VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYHLEDÁVÁNÍ V MULTIMODÁLNÍCH DATABÁZÍCH

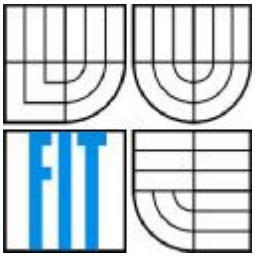
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. TOMÁŠ KREJČÍŘ

BRNO 2009

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYHLEDÁVÁNÍ V MULTIMODÁLNÍCH DATABÁZÍCH

MULTIMODAL DATABASE SEARCH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

BC. TOMÁŠ KREJČÍŘ

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR CHMELAŘ

BRNO 2009

Abstrakt

Práce se zabývá uložením a efektivním vyhledáváním velkých kolekcí multimediálních dokumentů. Tato oblast se nazývá vyhledávání informací. Multimediální dokumenty jsou reprezentovány vektory ve vysoko-dimenzionálním prostoru, neboť v takto modelované kolekci dokumentů lze jednodušeji definovat sémantiku i mechanismus samotného vyhledávání. Tento dokument popisuje řešení problému efektivního vyhledávání v kolekcích snímků, od extrakce rysů až po vyhledávání. Podrobněji pojednává o podobnostním vyhledávání založeném na metrickém prostoru, které využívá vzdálenostní funkce, jako např. Euklidovu, Chebyshevovu nebo Mahalanobisovu, pro porovnání globálních rysů a kosinovou větu pro porovnání lokálních rysů. Experimenty provedené na datové sadě TRECVid porovnávají implementované vzdálenostní funkce, ze kterých se pro globální rysy nejvhodnější ukázala Mahalanobisova vzdálenost a pro lokální rysy kosinová věta.

Abstract

The field that deals with storing and effective searching of multimedia documents is called Information retrieval. This paper describes solution of effective searching in collections of shots. Multimedia documents are presented as vectors in high-dimensional space, because in such collection of documents it is easier to define semantics as well as the mechanisms of searching. The work aims at problems of similarity searching based on metric space, which uses distance functions, such as Euclidean, Chebyshev or Mahalanobis, for comparing global features and cosine or binary rating for comparing local features. Experiments on the TRECVid dataset compare implemented distance functions. Best distance function for global features appears to be Mahalanobis and for local features cosine rating.

Klíčová slova

Vyhledávání informací, relevance, MPEG-7, multimediální databáze, extrakce rysů, indexování, metrický prostor, vzdálenostní funkce, TRECVid.

Keywords

Information retrieval, relevance, MPEG-7, multimedia databases, feature extraction, indexing, metric space, distance functions, TRECVid.

Citace

Bc. Krejčíř Tomáš: Vyhledávání v multimodálních databázích, diplomová práce, Brno, FIT VUT v Brně, 2009

Vyhledávání v multimodálních databázích

Prohlášení

Prohlašuji, že tato diplomová práce je mým původním autorským dílem, které jsem vypracoval samostatně pod vedením Ing. Petra Chmelaře.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Tomáš Krejčíř
25.5. 2009

Poděkování

Touto cestou bych rád vyjádřil poděkování vedoucímu práce Ing. Petru Chmelařovi za odbornou pomoc a praktické rady, za trpělivost při vysvětlování problematiky a za čas, který mi při vytváření této práce věnoval. Poděkovat bych chtěl i celé své rodině a přítelkyni, kteří mě po celou dobu studia podporovali.

Obsah

1 Úvod

Vždy byla potřeba nějakým způsobem ukládat či schraňovat nějaké informace, ať už textové nebo obrazové a vyhledávat v těchto materiálech. Vyhledávání informací, anglicky Information Retrieval (IR), je od dob vynálezu počítače obor, který jde stále kupředu, ale stále existuje mnoho nevyřešených problémů, ať už s uložením dat (především pak multimediálních) nebo se samotným vyhledáváním informací.

Čím dál tím více oborů lidské činnosti je spjato s informačními technologiemi a uživatelé informačních technologií vyvíjí velký tlak na rozvoj databázových systémů. Každým rokem se vyprodukuje obrovské množství dat v nejrůznější podobě, např. digitální fotografie, audio a video záznamy, snímky lékařských přístrojů, obrázky či různé dokumenty, která jsou v digitální formě ukládána do databází a následně po databázi požadována. Aby byla data snadno přístupná, musí být nějakým způsobem strukturovaná a musí dovolovat efektivní manipulaci. Z tohoto důvodu vznikly databázové systémy, které umožňují vhodnou organizaci dat a snadnou manipulaci s nimi jako je vyhledávání, vkládání, mazání nebo přesouvání.

Tradiční způsoby vyhledávání hledají množinu objektů přesně odpovídající zadanému dotazu, ovšem pro typy dat, jako jsou obrázky, audio a video záznamy, dokumenty nebo jejich kombinace, má přístup vyhledávání na přesnou shodu jen malý význam. U takových dat nás více než přesná shoda zajímají podobné objekty a touto oblastí se zabývá tzv. podobnostní vyhledávání. Vhodným matematickým modelem podobnostního vyhledávání je metrický prostor. Z každého objektu jsou vyextrahovány určité vlastnosti a na základě nich je vytvořena bodová reprezentace objektu v metrickém prostoru. Vzdálenost mezi body v metrickém prostoru pak představuje míru podobnosti mezi skutečnými objekty. Vzdáleností funkce nám zaručuje, že objekty, které jsou si podobné ve skutečnosti, jsou si blízké i v metrickém prostoru. Tento přístup dovoluje data organizovat a vyhledávat v nich.

Uplatnění podobnostního vyhledávání se nachází v mnoha oblastech reálného světa jako např. vyhledávání multimediálních informací, rozpoznávání obrazu, biomedicínské databáze či dolování dat. Důležitým požadavkem kvalitního návrhu podobnostního vyhledávání je výkon. Proto se objevují stále nové metody.

Ve druhé kapitole jsou uvedeny základní informace spolu s popisem typického systému vyhledávání informací a jejich modely, které vysvětlují strukturu a procesy takových systémů. Důležitým pojmem jakéhokoliv vyhledávání, který je také vysvětlen v této kapitole, je pojem relevance, který udává jak správné a jak užitečné výsledky byly procesem vyhledávání vráceny.

Třetí kapitola se zaměřuje na oblast vyhledávání v multimediálních databázích. V této kapitole se nalézá podrobný popis standardu MPEG-7 (pro popis multimedií), na kterém je z velké části postavena výsledná aplikace, dále zde čtenář najde popis extrakce rysů, což je první a nejdůležitější krok procesu vyhledávání a popis samotného vyhledávání. Tento popis je dosti obecný a zobrazuje abstraktní reprezentaci možné aplikace využívající MPEG-7.

Pokud se podíváme na vývoj multimediálních databází z hlediska objemu dat, často musíme manipulovat a přistupovat k více jak milionům, někdy až miliardám položek. V takovýchto případech je potřeba urychlit přístup k datům a jejich vyhledávání, což lze vyřešit indexováním. Popis indexových struktur lze nalézt na konci třetí kapitoly.

Problematice spojené s metrickým prostorem jako vyhledávání se podrobně věnuje čtvrtá kapitola. V této kapitole jsou popsány vzdálenostní míry (metriky), které byly použity pro výslednou aplikaci, dotazy na podobnost, které se běžně používají a nebo základní principy dělení, které jsou důležité pro indexaci.

Pátá kapitola se zabývá samotnou aplikací, resp. vývojem všech komponentů výsledné aplikace. Jsou zde popsány jednotlivé kroky procesu vyhledávání a u každého kroku je uveden popis, kdo se na něm podílel a v jakém programovacím jazyce byl implementován.

V šesté kapitole se věnuji experimentům a konkrétním výsledkům. Čtenář zde nalezne popis postupu, jak byly experimenty prováděny s následnými pěti experimenty a na závěr nechybí celkové zhodnocení všech experimentů.

2 Vyhledávání informací (IR)

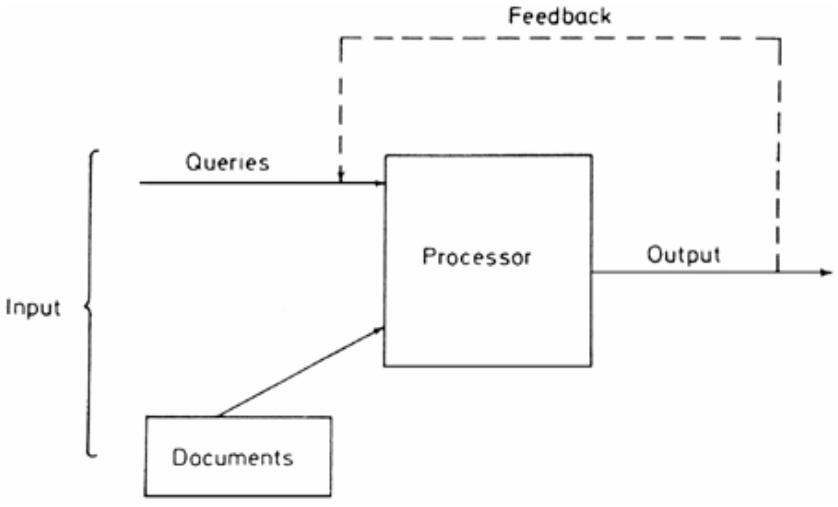
Vyhledávání informací (Information retrieval - IR) je velmi široký termín. Základní definice tohoto pojmu je taková, že jde o vyhledávání v nestrukturovaných datech. Mezi nejčastější vyhledávání informací patří vyhledávání v textových dokumentech jako například vyhledávání ve článcích, v novinách nebo vyhledávání po internetu. Mezi další typy vyhledávání patří rozpoznávání obrázků, rozpoznávání videa či rozpoznávání hudby. Vyhledávání informací je činnost, jejímž cílem je identifikace relevantních dokumentů nebo informací v informačních zdrojích, souvisí s reprezentací, skladováním, organizací a přístupem k informacím. Systémy vyhledávání informací v reálném světě jsou hodnoceny podle „spokojenosti uživatele“ a ceny, kterou je uživatel ochoten zaplatit.

Vyhledávání dat a vyhledávání informací je dosti odlišné a to především v otázce shody. U vyhledávání dat běžně hledáme naprostou shodu, tedy kontrolujeme zda porovnávaný vzorek je naprosto shodný s porovnávaným. U vyhledávání informací to může být stejné, avšak většinou se pokoušíme najít ty, které se částečně shodují a z nich vybereme několik nejlepších. Rozdíl je také v modelu, který je u vyhledávání dat deterministický na rozdíl od pravděpodobnostního modelu u vyhledávání informací. Další rozdíly lze najít např. v [48].

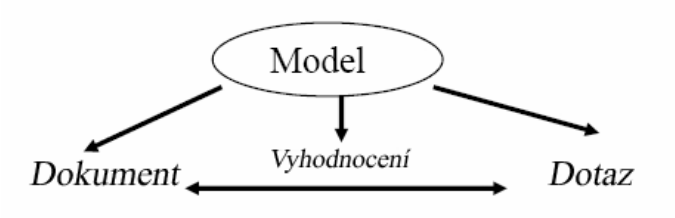
Od poloviny 19. století problém ukládání informací a vyhledávání v nich přitahuje větší a větší pozornost. Je to především tím, že máme obrovské množství informací, ke kterým chceme přesně a rychle přistupovat a to je čím dál tím těžší. Některé problémy vyhledávání jako například *uskladnění* informací byly vyřešeny počítači, nicméně problém efektivity vyhledávání zůstal nevyřešen. Když se staly dostupnými výkonné počítače, myšlenky byly takové, že počítač bude schopen „přečíst“ veškeré dokumenty a vyextrahovat jen ty relevantní. To se také brzy stalo použitím textového přirozeného jazyka dokumentu, ale zůstal nevyřešen problém jak charakterizovat obsah dokumentu. „Čtení“ zahrnuje pokusy vyextrahovat informace z textu, a to jak syntaktické tak sémantické, a využití těchto informací k rozhodnutí, zda každý dokument je relevantní nebo není. Obtížné není jenom to, jak extrahovat informace, ale také jak tyto informace použít k rozhodnutí relevance.

Rozumově je pro člověka možné zjistit relevanci dokumentu vzhledem k dotazu, avšak pro počítač potřebujeme udělat model, který dokáže spočítat rozhodnutí relevance.

Na obr. 2.1 lze vidět, jak vypadá typický *systém vyhledávání informací*. Takový systém se skládá ze tří hlavních komponentů – Input (vstup), do kterého patří Queries (dotazy) a Documents (dokumenty), procesor a Output (výstup). U on-line systémů je součástí i tzv. Feedback neboli zpětná vazba.



Obr 2.1 Typický systém vyhledávání informací [48].



Obr 2.2 Schéma modelu vyhledávání informací.

Hlavním problémem vstupní komponenty je získat reprezentaci všech dokumentů a dotazů vyhovující pro počítačové zpracování. Reprezentace dokumentu by mohla být extrakce slov, která jsou považována za podstatná. Pokud je systém on-line, je možné, aby uživatel měnil své požadavky během vyhledávání. Taková procedura je běžně zvaná jako ***feedback*** (zpětná vazba). Procesor je část systému, který se týká procesu vyhledávání. Proces může vyžadovat strukturování informací vhodným způsobem, jako např. klasifikace.

2.1 Modely IR

Modely IR vysvětlují strukturu a procesy systémů vyhledávání informací a objasňují jejich obecnost jako protiklad k specifické charakteristice. Model IR dává odpověď pro mechanizmus rozhodování relevance. Modely IR odpovídají na otázky relevance dotazu k dokumentům v databázi: „Jaké dokumenty mají být výsledkem dotazu?“ a „Jaké bude jejich uspořádání pro prezentaci uživateli?“.

Mechanizmy na rozhodnutí relevance mohou být buď přesné nebo flexibilní a reprezentace dat může mít proměnný stupeň abstrakce. Mezi klasické modely patří ***booleovský model***, ***vektorové modely*** či ***pravděpodobnostní modely***. Mezi alternativní modely patří modely neuronová síť, SVM či Fuzzy množina.

V dnešní době žádný skutečně dobrý vyhledávač nepracuje na základě jediného z modelů. Spíše se hledají cesty, jak několik modelů sladit tak, aby se jejich negativa vzájemně potlačila [26].

2.1.1 Booleovský model

Tento model je jeden z nejstarších vůbec a v minulosti býval hojně používán v knihovnických informačních systémech. Na dotaz vrací jako výsledky ty dokumenty, které obsahují slova z dotazu. V základní variantě neumožňuje stanovování relevance a funkce podobnosti je vyčíslována pouze s hodnotami 0 (zásah nenalezen) nebo 1 (zásah). V pozdějších variantách byl booleovský model obohacen o jemnější výpočet relevance.

Model předpokládá, že dokument *d* je popsán množinou klíčových slov. Binární rozhodovací kritérium pro tento model je založené na přítomnosti, resp. absenci daného klíčového slova a neuvažuje jeho váhu (pravdivost, dokazatelnost). Všechny dotazy jsou tvořeny termy a logickými spojkami *and, or, not* a závorkami. Výhodou tohoto modelu je efektivnost, mezi nevýhody patří to, že výsledkem je příliš málo dokumentů nebo příliš rozsáhlé odpovědi a neřeší uspořádání.

Několik typů binárních porovnání:

- *jednoduché*: $M = |X \cap Y|$, (2.1)
- *Diceovo:* $M = \frac{|X \cap Y|}{|X| + |Y|}$, (2.2)

- *Jaccardovo:* $M = \frac{|X \cap Y|}{|X \cup Y|}$, (2.3)
- *Kosinové:* $M = \frac{|X \cap Y|}{\sqrt{|X|} \cdot \sqrt{|Y|}}$, (2.4)
- *Overlapovo:* $M = \frac{|X \cap Y|}{\min(|X|, |Y|)}$. (2.5)

2.1.2 Vektorový model

Vektorový model umožňuje jemnější výpočet relevance. Myšlenka je opět velice jednoduchá, viz [26]. Každý text či dotaz je reprezentován bodem v n-rozměrném souřadnicovém systému. Tento bod představuje i vektor (začínající v počátku souřadnic), a tak dostal model i svůj název. Přesto, že vektorový model patří k nejstarším, není dosud známa efektivní implementace.

Konstrukce bodů je taková, že čím blíže jsou k sobě, tím více reprezentují podobný (dokonce téměř totožný) text. Dále ale budu hovořit o vektorech, nikoliv bodech. Skalární součin dvou vektorů vychází největší, pokud mají tyto vektory stejný směr. Jako nulový vychází, pokud mají vektory směr opačný. Tohoto jevu bývá s úspěchem využito pro vlastní kalkulaci podobnosti.

Předpokládejme, že v textech máme *n* rozdílných slov. Toto *n* také určuje *n*-rozměrnost našeho vektorového systému. Každý vektor pak na souřadnici - odpovídající danému slovu - obsahuje jeho četnost (ať již v jednotlivém dokumentu nebo třeba dotazu).

Podobnost stanovíme jako *skalární součin dvou vektorů* - ty budeme pro přehlednost označovat jako *w()*. V případě podobnosti dotazu a dokumentu pak tato formule vypadá takto:

$$sim_{w(d_j),w(q)} = \sum_{i=1}^{n} (w_{i,j} \cdot w_{i,q}), \quad (2.6)$$

kde $w(d_j) = (w_{1,j},..w_{n,j})$ a $w(q) = (w_{1,q},..w_{n,q})$. Hodnoty $w_{i,j}$ udávají počet slova t_i v j-tém dokumentu. Podobně $w_{i,q}$ udává počet slova t_i v dotazu q. V pokročilých implementacích již $w_{i,j}$ nepředstavuje četnost, ale naopak důležitost, která má většinou výchozí kalkulaci v četnosti.

Dále nebývá vhodné nechat růst vektory (jejich délku) nade všechny meze. Proto je výhodné *normalizovat* používané *vektory* na jednotkovou délku. Tím je pak do značné míry zajištěno, že původně dva krátké vektory (protože obsahovaly jen malý počet slov), jdoucí týmž směrem, nebudou při výpočtu podobnosti pomocí skalárního součinu přebity dvěma velice dlouhými vektory, které již ale nejdou tak úplně týmž směrem. Tak dostáváme základní formuli:

$$sim_{w(d_j),w(q)} = \sum_{i=1}^{n} \frac{w_{i,j} \cdot w_{i,q}}{length(w(d_j)) \cdot length(w(q))}. \quad (2.7)$$

Popíši i způsob, kterým lze efektivněji kalkulovat $w_{i,j}$. Četnosti slov, které jsme původně používali jako $w_{i,j}$, označme nyní $freq_{i,j}$ (frekvence i-tého slova v j-tém dokumentu) a počet slov dokumentu *j* označme $|freq_j|$. Pak jako normalizovanou frekvenci použijeme:

$$tf_{i,j} = \frac{freq_{i,j}}{|freq_j|}. \quad (2.8)$$

Dále spočteme *inverzní frekvenci dokumentu* pro i-té slovo:

$$idf_i = \log(\frac{N}{n_i}), \quad (2.9)$$

kde n_i je počet dokumentů obsahujících i-té slovo a *N* je počet všech dokumentů.

Pak můžeme určit $w_{i,j}$ takto:

$$w_{i,j} = tf_{i,j} \cdot idf_i . \tag{2.10}$$

Na závěr definuji *kosinovou větu*, podobnostní funkci, která klasifikuje podobnost vektoru dokumentu d_j k vektoru dotazu q (kosinus odchylky vektorů):

$$sim_{\cos(q,d_j)} = \frac{\sum_{i=1}^{n} q_i \cdot w_{i,j}}{\sqrt{(\sum_{i=1}^{n} q_i)^2 \cdot (\sum_{i=1}^{n} w_{i,j})^2}} . \tag{2.11}$$

Pomocí takové podobnostní funkce jsou realizovány rozsahové dotazy či dotazy *k* nejbližších sousedů, které jsou popsány v sekci 4.3.

2.1.3 Pravděpodobnostní model

V *pravděpodobnostním modelu* je rámcem pro modelování dokumentů a dotazů teorie pravděpodobnosti. Idea tohoto modelu je taková, že pokud se na určitý dotaz podaří vrátit výsledné dokumenty v pořadí s klesající pravděpodobností relevance, bude efektivita systému nejlepší možná. Nejlepším řešením tohoto modelu je potom Bayesův teorém:

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)} . \tag{2.12}$$

2.2 Měření efektivity

Důležitým pojmem při vyhledávání informací je *relevance* (neboli významnost či důležitost). Je to velmi subjektivní pojem a také měřítko toho, jak silně nějaká věc ovlivňuje realitu, jak kvalitní je informace nebo jak kvalitní je určitá teorie. Relevance je důležité kriterium kvality informace [48].

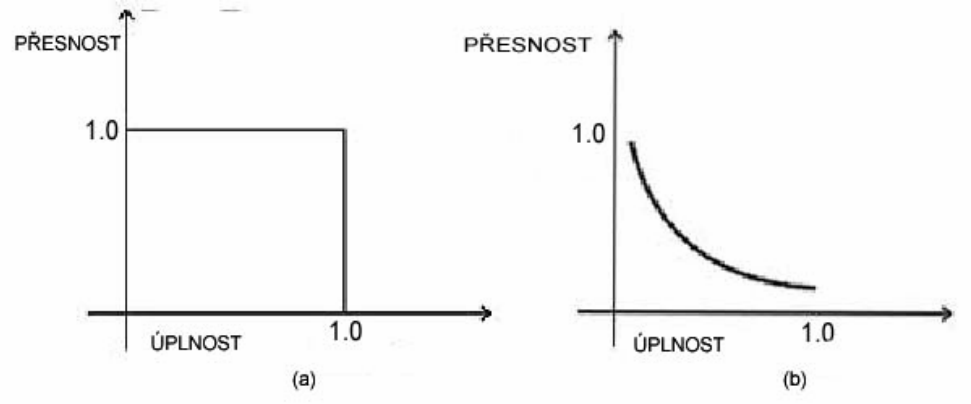
Významnost velmi silně závisí na kontextu a dává objektům určitou *váhu* v závislosti na vztazích mezi objekty. Čím vyšší je váha objektu, tím více je potřeba stav objektu sledovat, neboť jeho změny významně působí na okolí. To, co může být v jednom kontextu relevantní, může být v jiném kontextu zcela nevýznamné.

Určování důležitosti objektů se provádí určením důležitostí konkrétního subjektu a může být prováděno objektivními nebo subjektivními metodami. Objektivní metody spočívají v tom, že se určí konkrétní parametry a rozsah povolených hodnot. Tyto parametry se následně úplně a jednoznačně zaznamenají. Pokud není možné zjistit jednoznačně a přesně hodnoty parametrů, provádí se vyhodnocení více nebo méně intuitivně. Tzn. že vyhodnocení je založeno na znalostech, zkušenostech či osobních dojmech. Taková metoda je potom subjektivní.

U relevance se předpokládá, že dokument je nakonec předurčený k tomu, aby byl nebo nebyl relevantní podle uživatele. Efektivita je pouze míra schopnosti systému uspokojit uživatele v termínech relevance získaného dokumentu.

Existuje mnoho měr, které mohou být použity k určení přibližné kvality, jsou to např.:

- *Přesnost (*precision) – vyjadřuje, jak velká část nalezených dokumentů je relevantní
- *Úplnost* (recall) – vyjadřuje, jak velká část relevantních dokumentů (ze všech možných) byla vyhledána



Obr 2.3 (a) ideální poměr, (b) reálný funkční vztah mezi přesností a úplností [28].

Nechť *TP* představuje relevantní část výsledkové množiny dotazu, kterou vrátí proces vyhledávání, *T* je kompletní výsledková množina dotazu a *P* je relevantní část výsledkové množiny dotazu, kterou by měl vrátit proces vyhledávání. *Přesnost* je definována jako počet relevantních dokumentů získaných vyhledáváním podělených celkovým počtem dokumentů získaných hledáním:

$$Precision = \frac{|TP|}{|T|}. \tag{2.13}$$

Úplnost je definována jako počet relevantních dokumentů získaných vyhledáváním podělených celkovým počtem existujících relevantních dokumentů, které by měly být vráceny:

$$Recall = \frac{|TP|}{|P|}. \tag{2.14}$$

Určení přesnosti výsledku je poměrně jednoduché (za předpokladu, že výsledek je rozumně velký, hodně tedy záleží na použité vyhledávací strategii), obtížnější je to s určením úplnosti. Problém spočívá ve stanovení velikosti množiny všech relevantních objektů v prohledávané množině dokumentů [28].

Pro úspěšné vyhledávání je důležitá jak vysoká přesnost, tak vysoká úplnost. V ideálním případě by obě charakteristiky měly být rovny jedné (tj. každý relevantní dokument by byl nalezen a současně by výsledek byl tvořen pouze relevantními dokumenty), což je však nedostižný ideál. Reálně tento stav nikdy nenastává, neboť ***přesnost*** a ***úplnost*** jsou navzájem ***protichůdné***. Jestliže je dotaz formulován tak, aby poskytoval co nejúplnější výsledky, pak současně bude jeho výsledek málo přesný (relevantní objekty budou v menšině oproti nerelevantním). A naopak – pokud jsou výsledky díky vhodné formulaci dotazu velmi přesné, je velmi pravděpodobná nízká úplnost. Na *obr. 2.3 (a)* si lze všimnout ideálního poměru mezi přesností a úplností a na *obr. 2.3 (b)* je zobrazen reálný funkční vztah mezi těmito mírami.

Poslední mírou, o které se zmíním bude *Průměrná přesnost úplnosti a přesnosti*, popsána v [40]. Je definována takto:

$$AP = \frac{\sum_{r=1}^{N} (\mathrm{Pr}\,ecision(r) \cdot TP(r))}{P}, \tag{2.15}$$

kde *r* je pozice, *N* počet získaných dokumentů, *TP()* počet relevantních dokumetů dané pozice, *Precision()* je přesnost výřezu dané pozice a *P* udává počet relevantních dokumentů, které měly být získány.

3 Multimediální data

V dnešní době je obrovské množství dat a aplikací pracujících s rozmanitými daty, které požadují databáze. Je to především proto, že databáze zaručují konzistenci, integritu, souběžnost, bezpečnost a dostupnost dat. Z hlediska uživatele, databáze poskytují funkce pro jednoduchou manipulaci, dotazování a získávání relevantních informací z obrovských kolekcí uložených dat.

Multimediální databází může být např. kolekce obrázků, audia, videa, časových řad, textů, XML, atd. Obecně je to kolekce nestrukturovaných dat, u kterých vnitřní struktura i sémantika je skrytá a nejednoznačná - závislá na aplikaci, datech, i subjektivitě uživatele. U relačních databází se využívají dotazy na shodu, což u multimediálních databází nestačí.

V dnešní době jsou multimediální databáze prostě nepostradatelné. Biometrické databáze jsou obrazové databáze a využívají se především u policie na identifikaci osob. Video kolekce se využívají na záznamy z bezpečnostních kamer (letiště, supermarkety, centra měst, atd.) či filmové kolekce a zvukové databáze se využívají např. při zpracovávání řeči [16].

Mezi multimediální data patří např. zvukové stopy (audio), video záznamy či obrazová data, ale také mezi tato data patří např. strukturované texty či rukou psané poznámky. Můžeme tedy říci, že multimediální data jsou nestrukturovaná data, která dělíme na typy vizuální a audio. Tato práce se však dále zabývá pouze vizuálními daty.

Vyhledávání je stěžejní problém multimediálních dat. Vyhledání čísla, slova, případně názvu souboru je mnohem jednodušeji představitelné a realizovatelné, než vyhledání požadovaného objektu na obrázku, akce ve videu, slova nebo melodie ve zvukových datech. Vyhledávání v multimediálních datech se tedy děje výhradně pomocí metadat.

3.1 Metadata

Metadata jsou data o datech, která popisují obsah, kvalitu, stav a jiné charakteristiky (multimediálních) dat. Metadata nejen identifikují a popisují informační objekt, ale také dokumentují, jak se informační objekt chová, jeho funkce a užití, jeho vazby a vztahy s jinými informačními objekty a jak může být řízen, spravován apod. Metadata pomáhají člověku data nalézt a rozumět jim.

Digitální knihovny vyžadují přesný a strukturovaný popis každé uchovávané jednotky, avšak digitální knihovny nejsou jediné, které tuto vlastnost potřebují. Existuje několik typů metadat [24]:

- *Administrativní metadata* jsou data nezbytná pro uchování, správu a následné zobrazení obsahu. Obsahují například způsob a čas pořízení, kódování, formát a velikost obsahu, nebo informace o jeho zabezpečení. Jsou záležitostí daného formátu, případně multimediálního kontejneru. Databázový systém by měl umět tyto data interpretovat a využít především pro svou interní potřebu. Pro vyhledávání jsou spíše doplňkové.
- *Strukturální metadata* popisují interakci objektů. Například do kterého videa patří daná stopa nebo snímek, stránka dokumentu (kompozice), jejich kategorické uspořádaní (agregace), případně jejich provázání odkazy (asociace).
- *Popisná metadata* jsou informace popisující zdroj, jsou viditelná uživateli a slouží především pro vyhledávání.

Multimediální data obsahují v hlavičce souboru obvyklé informace jako rozlišení, typ, barevnost či kódování. Kromě těchto informací však mohou obsahovat ještě další informace, jako např. MPEG-7 nebo jiné, které lze najít např. v [16].

3.1.1 MPEG-7

MPEG-7 [přeloženo z 21] je standard ISO/IEC vyvinutý konzorciem MPEG (Moving Picture Experts Group), tedy tím, které stojí za úspěšnými standardy MPEG-1 (1992), MPEG-2 (1994), a MPEG-4 (1998 a 1999). Standardy kódování videa MPEG-1 a MPEG-2 jsou základem celého segmentu produktů a technologií, jako Video CD, MP3, DVD, digitální televize DVB aj.

Standard MPEG-7, formálně také zvaný "Multimedia Content Description Interface", je řadou standardizovaných nástrojů pro *popis multimediálního obsahu*. MPEG-7 nabízí komplexní sadu nástrojů pro audiovizuální popis (Description Tools jsou metadatové elementy, jejich struktura a vztahy, které standard definuje ve formě deskriptorů a deskripčních schémat). Ty jsou základem pro aplikace, kterým umožní efektivní přístup k multimediálnímu obsahu.

MPEG-7 poskytuje sadu standardizovaných deskriptorů pro popis obsahu různých druhů médií – statické obrazy v tištěné podobě, 3D grafika a její modely, zvuk, řeč a video nebo biometrie lidského obličeje – s cílem efektivního vyhledávání informací ve velkém množství multimediálních dat. Popis nesouvisí se způsobem uložení médií. Je možné přiřadit deskriptor MPEG-7 k obrazu, článku, filmu stejně jako proudu MPEG-4. Pro uložení schémat deskriptorů je definován formát založen na XML, který lze převést do efektivní binární podoby.

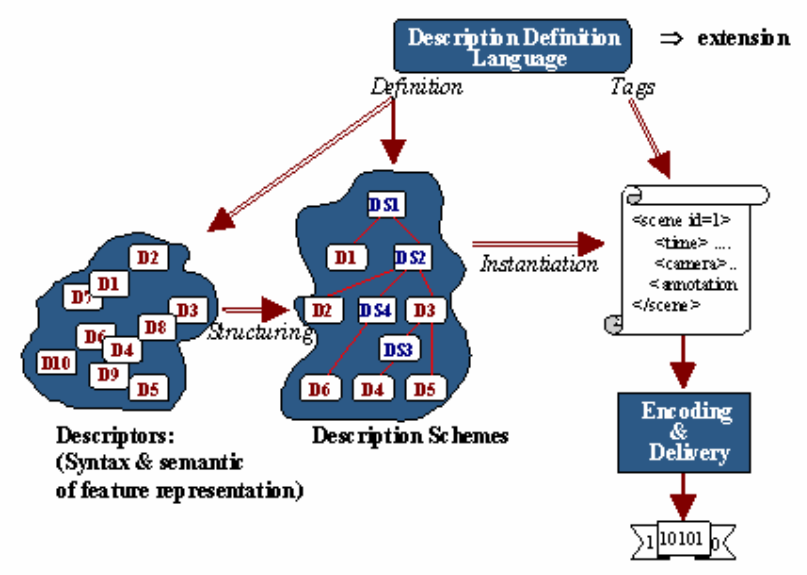
Základní elementy standardu MPEG-7 jsou [16]:

- *Deskriptory* (D) reprezentují vlastnosti, rysy – atributy multimediálního obsahu založené na katalozích (název, autor, práva, popis), sémantice (kdo, co, kdy a kde – informace o objektech a událostech), syntaxi (pro CBR, barva obrazu, tón zvuku) a technologii (formát, velikost, vzorkovací frekvence).
- *Popisová schémata* (Description Schemes – DS) popisují strukturu a sémantiku vztahů mezi komponentami D nebo DS – typ média, jeho původ, možnosti použití, strukturální vlastnosti nebo libovolný text.
- *Jazyk pro definici deskriptorů* (Description Definition Language – DDL) definuje D, DS, DT(Datové typy - zavádějí například jednoduchá pole v XML), jejich syntax, sémantiku, možnosti jejich změny a rozšíření založené na XML, upravené za pomoci W3C pro MPEG-7.
- *Systémové nástroje* (Systems tools) podporují tvorbu a přenos popisů například v binární podobě, jejich multiplexování s multimediálním obsahem, synchronizaci, formáty souborů.

Na obr. 3.1 můžeme vidět vztahy mezi elementy MPEG-7 zmiňovanými výše.

Standard MPEG-7 se skládá z následujících částí [21]:

1. *Systém*: specifikuje nástroje pro přípravu deskriptorů k přenosu, ukládání, kompresi a pro synchronizaci s obsahem.
2. *DDL*: specifikuje jazyk pro definování standardní množiny deskripčních nástrojů (DS, D, DT) a nových nástrojů, založeno na XML.
3. *Vizuální*: obsahuje nástroje pro popis vizuální složky, specifikuje základní kategorie – barva, textura, tvar, pohyb, pozice, rozpoznávání obličeje.
4. *Audio*: definuje nástroje pro popis zvukové složky jako spektrální, časové, dynamické vlastnosti (low-level) nebo rozpoznání hlasu, barva nástroje, melodie (high-level).
5. *Schémata popisující multimédia*: nástroje pro popis multimediální části popis obsahu – audio i vizuální (video), použití obsahu, organizace, navigace, interaktivity.
6. *Referenční software*: implementace standardu se stále vyvíjí, protože norma popisuje způsob uložení popisu, nikoli způsob jeho získání.



Obr 3.1 Schéma základních elementů MPEG-7 [21].

7. *Testování shody*: poskytuje vodítka pro testování shody implementace deskriptorů.
8. *Extrakce a použití MPEG-7 schémat*. Pouze informativní, není součástí normy, např. implementace experimentálního modelu.
9. *Profily a úrovně* – poskytují směrnice a profily standardu
10. *Definice schématu* – specifikuje schéma využívající DDL

3.2 Extrakce rysů

Existují dvě možnosti realizace popisu média a to ručně nebo automaticky. Ruční popis znamená vytvoření smysluplného názvu, popisu objektu popřípadě i ruční zařazení média do určité kategorie. Automatický popis vytváří *vektor rysů* pomocí procesu známého jako *extrakce rysů*. Rysy jsou charakteristické vlastnosti obsahu média. Rysy obrazů mohou být například barevnost, histogram či textury, rysy pohyblivých obrazů mohou být např. pohyb člověka. V oblasti rozpoznávání obrazu a zpracování obrazu je *extrakce rysů* speciální formou snížení dimenze.

Extrakce rysů je tedy automatizovaný proces redukce množství (nestrukturované) informace, jejímž výsledkem jsou (strukturovatelné a dále zpracovatelné) vektory rysů, popisující daný multimediální objekt. Vektorem rysů tedy rozumíme vektor číselných hodnot. Dále se budeme zabývat extrakcí rysů z vizuálních dat.

Extrakce nízko-úrovňových rysů obrazových dat jsou relativně jednoduché. Jde například o dominantní barvy v obrázku nebo její histogram. Pro některé rysy nejsou klasické barevné modely (RGB, CMYK) příliš vhodné. Je tedy vhodnější použít model, který více odpovídá lidskému chápání barev, jakým je YC_bC_r. Texturové vlastnosti reálných povrchů, 2D vizuální vlastnosti, jako jsou kontrast, zaměřenost, pravidelnost, nebo hmatový charakter, jako hrubost nebo drsnost. V její počítačové reprezentaci je složená z opakujících se elementů, které se nazývají primitiva. Tato primitiva mohou být popsána jejich statickou, geometrickou nebo frekvenční charakteristikou [16].

Existují dva hlavní typy vizuálních rysů [17]:

- ***Globální rysy*** jsou vlastnosti celého snímku. Jejich příkladem je barevný histogram, lokalizovaný histogram, dominantní barva, její rozložení, texturní vlastnosti (celého obrázku), množství pohybu ve scéně a pohyb kamery.
- ***Lokální rysy*** umožňují vytvořit vhodnou reprezentaci pro nalezení objektů tak, jak je chápou lidé, složených pomocí objektů, které dokáží "pochopit" zase počítače. Extrakce lokálních rysů sestává ze dvou základních kroků. Prvním z nich je detekce "regionů zájmu", jako jsou hrany, rohy nebo (barevně) spojité oblasti, komunitou počítačového vidění označované jako regiony či bloby, obvykle ohraničené elipsou. Nejdůležitější vlastností detektoru je jeho opakovatelnost – schopnost stejných detekcí při různých fotometrických (světelných) a geometrických podmínkách (velikost, rotace, úhly pohledu), odolnost vůči šumu (a kompresi obrazu).

3.2.1 Extrakce tvaru

Prvotní problém u vizuálních dat je jejich přílišná rozmanitá. Vnímání objektů člověkem je odlišné od způsobu jejich rozpoznání počítačem, tedy oblasti s určitou texturou, barvou atd. mohou mít různou velikost nebo orientaci a přitom mají stejný význam nebo naopak. Pokud nebudeme přemýšlet nad stupněm vhodné granularity přesnosti, je možné říci, že objekt se skládá z jednoho nebo více regionů v obraze. Oblasti je možné popsat jako plochu nebo její hranici. V případě plochy bývá oblast reprezentována jako bitmapa, ve které hodnoty jasu jedna značí příslušnost odpovídajících oblastí, zatímco „0" směřuje k bílému pozadí. Tento způsob popisu oblastí je efektivní při vyhledávání v případě porušení hranic.

Reprezentace hranic je náročnější, ale lépe odpovídá lidskému vnímání a je více odolná vůči transformacím, nehledě na změnu tvaru objektu v čase [16].

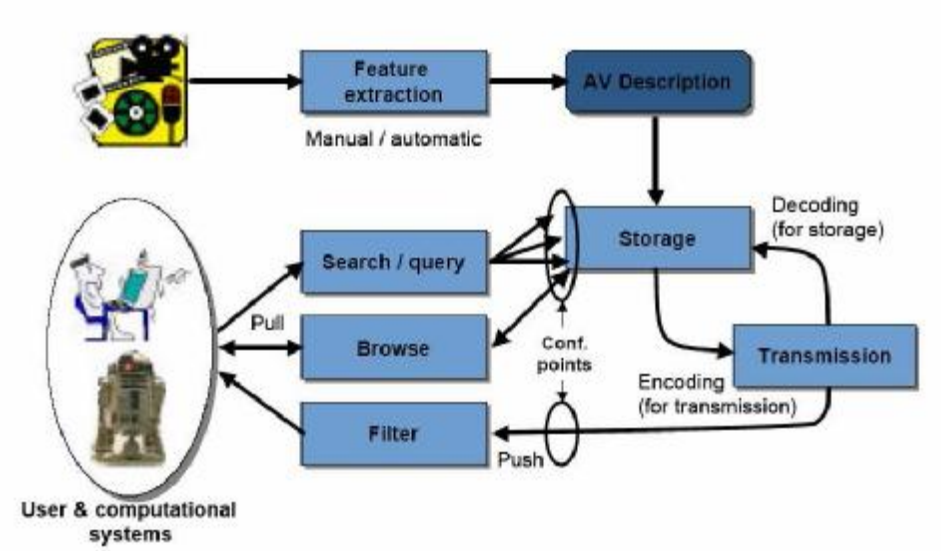
3.2.2 Extrakce pohybu a umístění

Pohybem míníme změnu umístění objektů v posloupnosti snímků, včetně dalších transformací objektu, jako zmenšení, otočení či deformace. Musíme počítat s přiblížením, zmenšením, otočením, perspektivní deformace nebo i s pohybem kamery. Kamera má šest stupňů volnosti, může se pohybovat po svislé nebo vodorovné ose, přibližovat nebo vzdalovat se a v každé z těchto os může navíc ještě rotovat. Z těchto důvodů pohyb kamery není vždy jednoduché přesně rozpoznat, ale je to žádaná vlastnost. Pohyb kamery můžeme vyjádřit pomocí fundamentální matice, kterou odhadneme díky shodě několika náhodných bodů v pokračujících snímcích. Když se nám podaří kompenzovat pohyb kamery odečtením vektoru pohybu a ohraničíme pohybující se oblasti, jsme schopni identifikovat pohyblivé objekty a určit jejich tvar. Takový objekt následně obalíme konvexní obálkou a zapíšeme souřadnice do databáze. Pokud navíc ještě doplníme časový údaj, dostaneme časoprostorový lokátor objektu.

Zobecněním chápání pohybu objektů na ploše vytvoříme perspektivní matici. Pokud budeme uvažovat nejjednodušší pohyb objektu v rovině, kdy připustíme pouze transformaci a rotaci objektu, těleso nemění příliš vzhled, tvar ani velikost. Takovéto objekty můžeme ve videu najít pomocí odhadu pohybu, které realizují zjištění změny odečtením vždy dvou po sobě jdoucích snímků. Z této extrakce se pak vytvoří vektor pohybu [16].

3.3 Vyhledávání

Jako elementární úkol každého multimediálního databázového systému je vyhledávání relevantní informace v audiovizuálních datech. Požadujeme samozřejmě také co nejefektivnější vyhledávání, a



Obr 3.2 Abstraktní reprezentace možné aplikace využívající MPEG-7 [21].

proto je v naprosté většině případů omezeno na metadata (obsahující popis i rysy), která musí být předem extrahována a vhodně indexována, na čemž výrazně závisí efektivita vyhledávání. Vyhledávací algoritmus je také velmi důležitý, ale především musí umět interpretovat správně daná metadata.

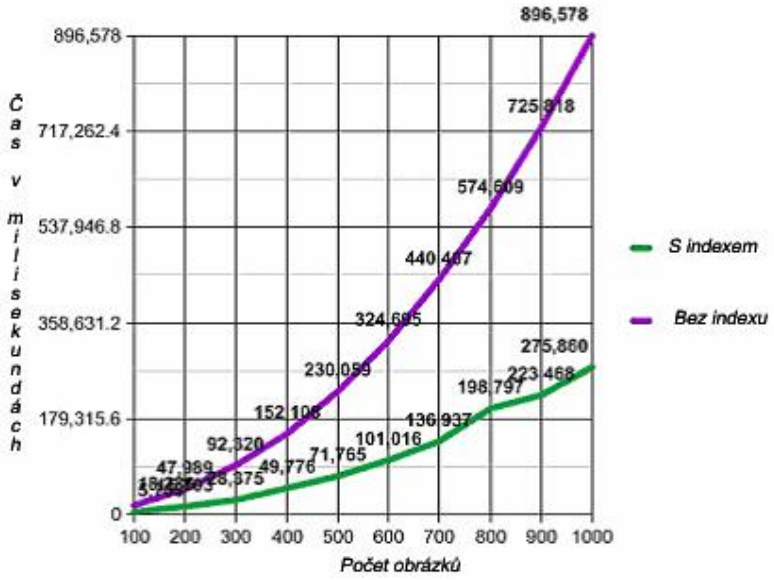
Metadata jsou relativně běžná, strukturovaná data a je tedy možné využít libovolný strukturovaný dotaz. Tak se tomu děje v případě vyhledávání dle textového popisu dat, které je v dnešní době nejběžnější, např. vyhledávání obrázků na internetovém portálu Gogole. Pro vyhledávání dle obsahu audiovizuálních dat je výše popsaný způsob dosti nepraktický. Například při dotazu na nalezení jisté textury by bylo nutné znát všech 30 koeficientů deskriptoru. Takový způsob se nazývá vyhledávání specifikací vlastností [16].

Ve vyhledávání zaměřeném na obsah se obvykle používají 2 typy dotazů:

- *Předložení vzorového obrazu* (similarity search) - využívá se porovnání deskriptorů extrahovaných z obrázku dotazu s deskriptory uloženými v databázi.
- *Specifikace vlastností* - prvky, ze kterých je tvořen deskriptor jsou porovnány s těmi v databázi.

Schéma a základní procesy multimediální databáze lze zjednodušeně pochopit z obrázku Obr 3.2, který zobrazuje abstraktní reprezentaci možné aplikace využívající MPEG-7. První a velmi důležitou součástí procesu je získání multimediálních dat, na kterých je celý proces postaven. Z multimediálního obsahu se získá audiovizuální popis (Feature extraction) a to pomocí manuální nebo poloautomatické extrakce rysů. Tyto extrakce tvoří deskriptor (AV Description), které jsou uloženy spolu s daty do databáze (Storage). Uživatelé či jiné služby mohou poté databázi dotazovat (Search / query) a výsledky těchto dotazů si prohlížet (Browse). Zobrazené výsledky lze filtrovat.

Důležitým pojmem této práce je *podobnostní vyhledávání*. Princip podobnostního vyhledávání předpokládá, že pro vyhledávání požadované informace jsme schopni dodat nějaký příklad požadovaného vizuálního obsahu. Z tohoto média multimediální databázové řízení báze dat extrahuje stejné rysy, jako by jej chtěl uložit do databáze a tato metadata dotazu jsou poté využita pro vyhledávání. Této oblasti se detailněji věnuji v kapitole 4.



Obr 3.3 Motivace využití indexu v databázi Oracle interMedia [11].

3.4 Indexování

Index je databázová konstrukce [47], sloužící ke zrychlení vyhledávacích a dotazovacích procesů v databázi. Je obvykle definován výběrem tabulky a jednoho konkrétního sloupce (nebo více sloupců), nad kterými si analytik nebo designér databáze přeje dotazování urychlit, dále pak technickým určením datového typu a typu indexu. Důležitá je skutečnost, že se indexují metadata. Chování a způsoby uložení indexů se mohou významně i výrazně lišit podle použité databázové technologie. Výjimku mohou tvořit například full-textové indexy, které jsou v některých případech definovány nad celou databází, nikoliv nad konkrétní tabulkou. Každá tabulka může mít indexů několik a indexovat je možné jak obsah tak i popis. Na obr. 3.3 je znázorněna motivace využití indexu při dotazování obsahu obrázkových dat.

Vytvoření indexu databázový server vyhradí pro požadovaný index určitou část paměťového prostoru a uloží do něj informace o rozmístění hodnot indexovaných sloupců v tabulce (obvykle ve strojové a pro člověka nečitelné podobě, která je závislá na vnitřních algoritmech indexace). Pokud později dojde k dotazu, který se týká indexovaných sloupců, není tabulka prohledávána podle toho, jak jsou za sebou řádky uloženy, ale pomocí informací uložených v paměťovém prostoru indexu je přistupováno přímo k relevantním řádkům tabulky.

Pro správné zvolení indexů by ten, kdo databázi navrhuje, měl vědět, jak často se vybírané záznamy z dané tabulky třídí podle zamýšleného sloupce (a kandidáta na index) a nakolik je důležité, aby vyhledávání a třídění podle něj bylo rychlé. U některých tabulek, které jsou např. číselníky o stálém počtu položek, řekněme max. několika desítkách, nemusí být třeba index definován vůbec.

Databázové indexy dělíme do různých druhů podle toho, co chceme při přístupech k primárním datům příslušné databázové tabulky optimalizovat. Označení druhů indexů se může různit, nejčastěji se používají tyto hlavní druhy [47]:

- *Primární index* tvoří sloupec (nebo více sloupců), který obsahuje primární klíč. Jedná se o zvláštní druh indexu, který se v každé tabulce může vyskytovat nejvýše jednou. Je definován sloupcem (sloupci) tabulky, který svou hodnotou jednoznačně identifikuje každý záznam.

- *Unikátní index* je tvořen ze sloupců obsahujících *unikátní klíč*, jedná se o speciální druh indexu, který je významově podobný předchozímu typu PRIMARY co do jednoznačnosti záznamu podle unikátní hodnoty klíče (typ PRIMARY je pouze zvláštní případ typu UNIQUE, jedná se vlastně o podtyp) v databázové tabulce (ostatně, jak naznačuje i sám jeho název) a v praktickém dopadu, který ho pak na práci s příslušnou databází má. Na rozdíl od předchozího typu však nemusí být unikátní index jediný, ale může jich být definováno více.
- *Index* též zvaný SECONDARY je tvořen pro sloupce, které obsahují *sekundární klíč* čili *druhotný klíč*. Definicí jednoho či více indexů tohoto typu v tabulce zajišťujeme optimalizaci vyhledávání podle dalších sloupců, mimo primární nebo unikátní indexy.
- *Full-text* - vytvořením indexu tohoto typu se databázový server bude snažit optimalizovat full-textové vyhledávání v daném sloupci u dané tabulky.
- *Parciální index* je index, který spadá do jednoho z výše uvedených základních druhů (v drtivé většině případů INDEX), ale týká se jen určité podmnožiny řádků tabulky. Databázový stroj s ním pracuje jen za určité podmínky. Tato podmínka se týká některých z ostatních polí v dané tabulce, tato podmínka je součástí definice parciálního klíče. Parciální klíče mají jen některé databáze (např. PostgreSQL). Situace, kdy záznamy tabulky příliš nerovnoměrně spadají do jedné velké podskupiny, je příkladem a důvodem pro vytvoření parciálního indexu, který v této podskupině (a pouze v ní) záznamy zindexuje.

Pro dynamickou indexaci primárního klíče u relačních databází se využívá stromů a různých variant (B+, B*), kdežto pro indexaci sekundárních indexů používáme různá vektorová řešení organizace indexového prostoru, abychom byli schopni odpovědět na různé variace dotazů.

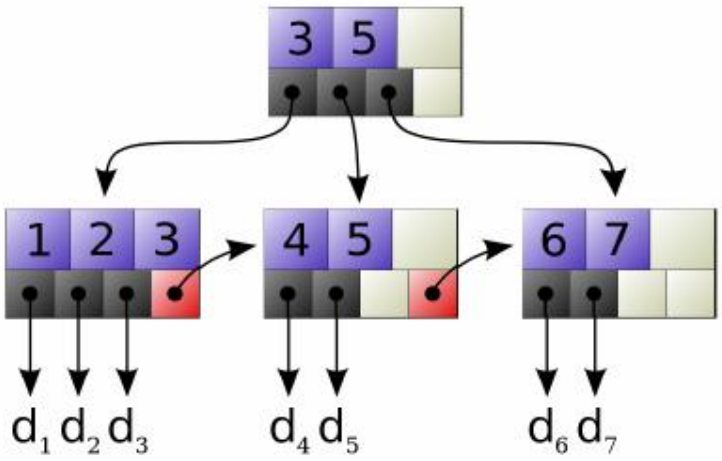
Tradičním požadavkem multimediálních databází je indexování (anotovaných) objektů. Problémem multimediálních dat je automatické nalezení těchto objektů a především jejich vícerozměrná povaha. Nelze tedy použít klasické metody, ale je možné využít struktury založené na dělení prostoru, které jsou popsány v sekci 4.4.

3.4.1 B-strom

B-strom je druh stromu (acyklický graf s jedním kořenem a vrcholy), který má řád *n* a definuje limity na maximální a minimální počet potomků vrcholu. Díky této vlastnosti je B-strom vyvážený a operace přidání, vyjmutí a vyhledávání probíhají v logaritmickém čase. Tato struktura je často používána v aplikacích, kdy není celá struktura uložena v paměti, ale v nějaké sekundární paměti, například databázi. Protože přístup do tohoto typu paměti je náročný na čas (hlavně vyhledání náhodné položky), snažíme se minimalizovat počet přístupů do této paměti. B-stromem řádu *n* rozumíme strom, splňující tyto vlastnosti [46]:

- Kořen má nejméně dva potomky, pokud není listem.
- Každý uzel kromě kořene a listu má nejméně $\left\lceil \frac{n}{2} \right\rceil$ a nejvýše *n* potomků.
- Každý uzel kromě kořene má nejméně $\left\lceil \frac{n}{2} \right\rceil - 1$ a nejvýše *n-1* položek.
- Všechny cesty od kořene k listům jsou stejně dlouhé.

Existuje mnoho variací tohoto stromu jako např. dosti používaný B+ strom, který má všechna data uložena až na samém konci stromu, tedy v listech a ostatní vlastnosti jsou stejné jako B-strom.



Obr 3.4 Ukázka B+ stromu [46].

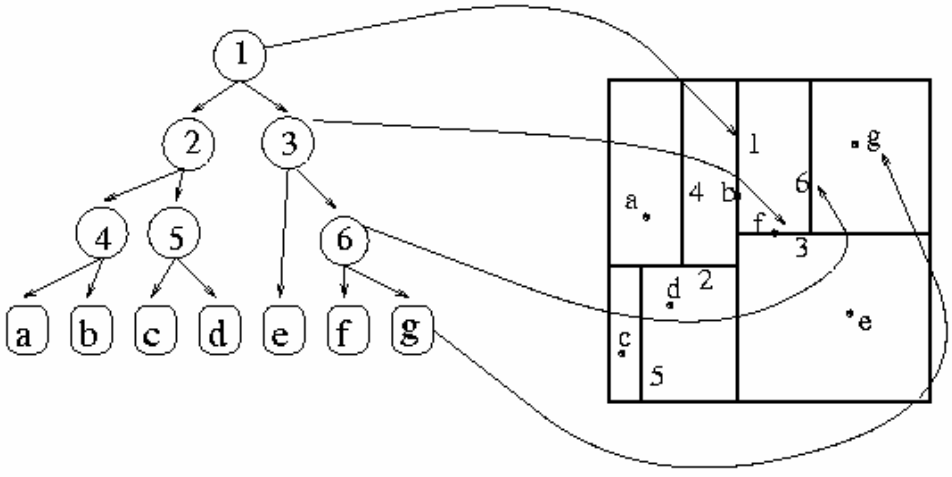
Ukázka B+ stromu je zobrazena na obr. 3.4 a výhodou tohoto stromu je velice rychlé získávání (čtení) souvislého bloku dat. Další modifikací je pak B* strom, který omezuje spodní hranici potomků.V B* stromu řádu *n* musí mít všechny uzly ve stromu mimo kořen minimálně *2/3*n* dětí.

3.4.2 K-dimensionální strom (K-D strom)

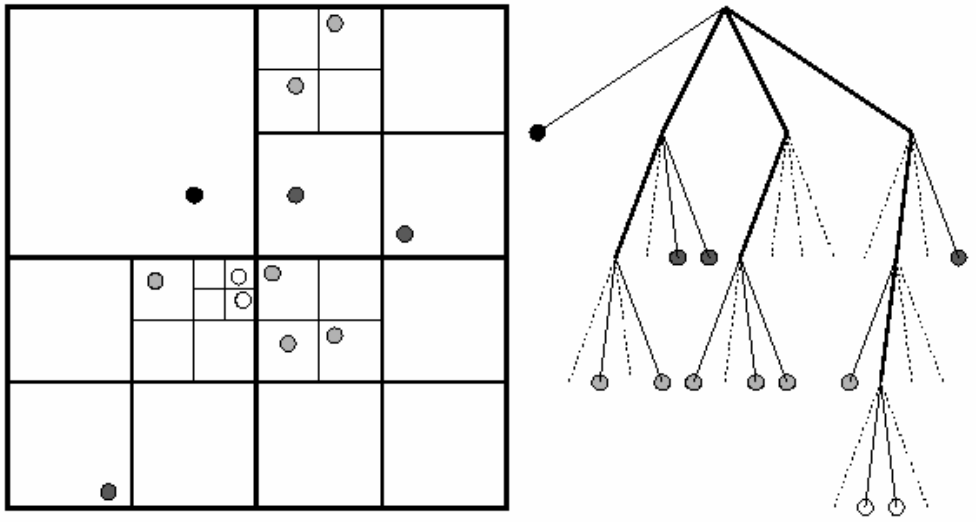
Prohledávání prostoru obecně mívá vysokou výpočetní složitost. Efektivnější vyhledávání lze dosáhnout vymezením podprostoru a to např. pomocí nadrovin (rovnoběžných s příslušnými osami), vymezit část k-rozměrného prostoru, která buď obsahuje přímo hledané řešení nebo umožňuje přijatelné libovolné dohledání. K-D stromy slouží pro ukládání bodových k-dimenzionálních dat.

Prostor je rozdělován tak, že se za *bod dělení* (jímž prochází příslušná nadrovina) volí *medián* souřadnic bodů v příslušném podintervalu vzniklém rozdělením předchozího intervalu. Tento postup vytváří vyvážený strom, což lze vidět na obr. 3.5. Nevýhoda K-D stromů je špatné pořadí při vkládání, výhodou je snadná implementace.

Jde o elegantní a intuitivní algoritmus s dobrým výkonem, vhodné pro dotazy extrakcí, rozsahové a nejbližších sousedů. Existují různá rozšíření, jako KDBstrom, který dělí adresní prostor do intervalů pro každý uzel, nebo hBstrom. U řady modifikací je volen jiný bod dělení než medián, avšak pak nemusí být zaručena vyváženost stromu.



Obr 3.5 Dělení 2D prostoru pomocí kd-stromu.



Obr 3.6 Dělení 2D prostoru pomocí Quad-stromu [25].

3.4.3 Quad strom

Quad strom je metoda ukládání a vyhledávání záznamů v databázi. Algoritmus vyhledává data opakovaným dělením záznamů do 4 částí, dokud nezůstane pouze jeden. Vychází z K-D stromu, ale vkládané body rozdělují prostor ve všech směrech, viz obr. 3.6. Quad stromy umožňují oproti K-D rychlejší vyhledávání, ale operace rušení je náročnější, kvůli nalezení uzlu, který má být nahrazen.

Jejich modifikací jsou MX-quad stromy. Také dělí prostor podle všech souřadnic, ale jen tak, aby vzniklé části byly stejně velké. Dále dělí rekurzivně. V těchto stromech je pevně určena granularita, problémem je její efektivní určení.

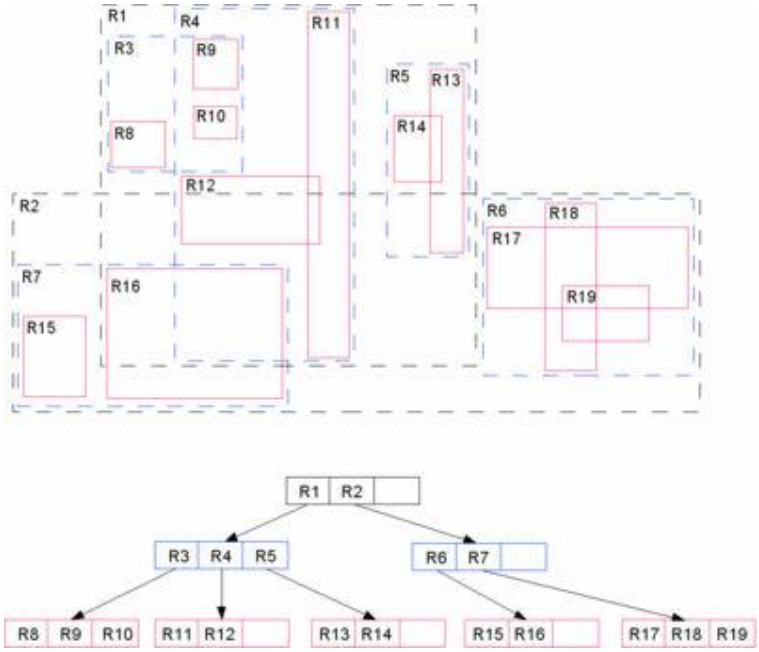
Zobecnění K-D a Quad stromu je BSP strom, který obdobně rozděluje prostor libovolnými rovinami na konvexní oblasti, obsahující maximálně jeden objekt, případně jeho část [16].

3.4.4 R-strom

R-stromy jsou datové struktury podobné B-stromům, ale využívají se např. na indexování multi-dimensionálních informací. R-stromy jsou založeny na ukládání obdélníkových struktur. Některé obdélníky představují reálné objekty vložené do databáze. Další sdružují menší obdélníky do větších, viz obr. 3.7. Všechny uzly stromu, mimo kořenového, musí být alespoň z poloviny zaplněné. Pro neobdélníkové objekty se hledá minimální ohraničující obdélník. Modifikací R-stromů jsou M-stromy s kruhovými (kulovými) oblastmi, což je výhodnější pro určování eukleidovských vzdáleností, ale výpočetně náročnější [16].

R-stromy jsou nejvíce použitelné pro velké množství dat. Hlavním problémem těchto stromů je možné překrytí obdélníků. Kvůli tomu nelze při vyhledávání vyloučit některé větve, ale je nutné prohledávat větší část stromu. Vylepšenou modifikací jsou R+ stromy a R* stromy. R+ stromy vylučují překrývání obdélníků, avšak obdélník může být obsažen v několika listech. Běžnější modifikace jsou R* stromy, které při výběru uzlu pro zařazení a štěpení snaží překrytí minimalizovat [33].

R-stromy historicky negarantovaly dobrý výkon, ale obecně fungují s reálnými daty dobře. Nicméně v roce 2004 byl publikován nový algoritmus, který definuje ***Přednostní R-Strom*** (Priority R-tree), který tvrdí, že je účinný jako doposud nejúčinnější metoda [45].



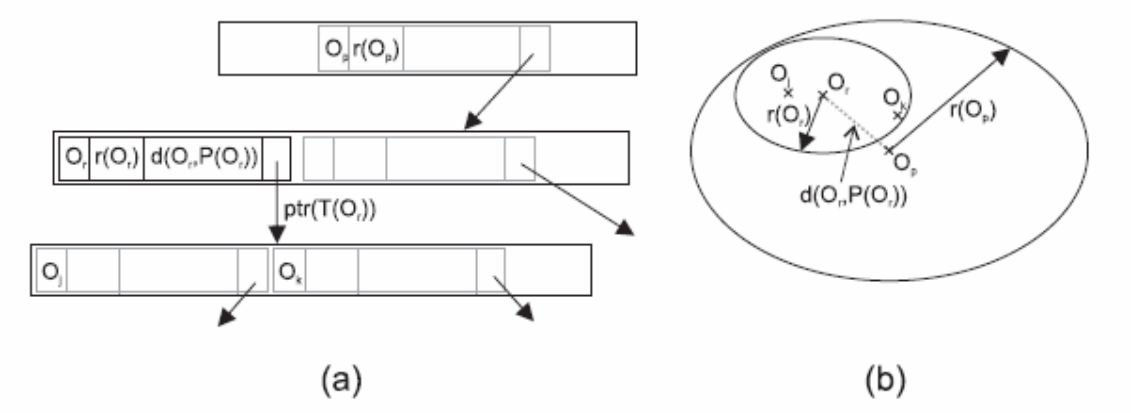
Obr 3.7 Dělení 2D prostoru pomocí R-stromu [45].

3.4.5 M-strom

Datová struktura M-strom představená v [35], se využívá pro indexování objektů vlastností metrických prostorů. Podobně jako mnoho struktur v jiných oblastech indexování i struktura M-stromu je založena na myšlence B+-stromu, tj. je vyvážená, dynamická a stránkovaná (umožňující efektivní perzistenci). Konkrétní index M-stromu představuje hierarchii metrických regionů (každý uzel představuje jeden region), respektive hierarchii shluků objektu v těchto regionech [29].

Indexované objekty jsou uloženy v listech stromu, nelistové uzly obsahují tzv. směrovací objekty. Záznam směrovacího objektu obsahuje samotný objekt, odkaz na svůj podstrom, hodnotu *r* (pokrývací poloměr) a vzdálenost od rodičovského směrovacího objektu.

Fascinující vlastností M-stromů je fakt, že jimi lze indexovat prakticky cokoliv, na čem může být definována metrika (resp. vzdálenost). Tato obecnost je na druhé straně vykoupena dost složitým managementem štěpení a slučováním regionů při základních operacích tak, aby zůstala zachována vysoká efektivita [30].



Obr 3.8 (a) Struktura M-stromu, (b) Odpovídající dělení metrického prostoru [19].

3.4.6 Invertovaný index

Invertovaný index (také označován jako invertovaný soubor) je indexová datová struktura ukládající zobrazení obsahu, jako slova nebo čísla na pozici v dokumentu. Invertovaný soubor je nejvíce populární datová struktura využita v systémech vyhledávání dokumentů. Základem jsou slova či fráze vybrané z dokumentů nebo označující jejich obsah a odkazy na jejich umístění v databázi. Podrobněji o tomto indexu lze nalézt v [12].

Zobecněný (invertovaný index - GIN) znamená, že index neví, která operace ho urychlí. Pracuje s určitými strategiemi, definovanými pro specifické datové typy. GIN (Generalized Inverted Index) je podobný jako GiST (Generalized Search Tree) a liší se od indexu B-stromu, který má předdefinované srovnávací operace. Detailněji o GIN v [37] a o GiST v [42].

Na závěr přidáme jeden příklad [18]. Máme tři dokumenty D_1, D_2 a D_3 a pozice slov v dokumentu:

D_0 = ‘It is what it is.’ ‘it’:0,3 ‘is’:1,4‘ what’:2
D_1 = ‘What is it?’ ‘what’:0 ‘is’:1 ‘it’:2
D_2 = ‘It is a banana.’ ‘it’:0 ‘is’:1 ‘a’:2 ‘banana’:3

Pak invertovaný index (GIN):

'a': {(2, 2)}
'banana': {(2,3)}
'is': {(0,1), (0,4), (1,1), (2,1)}
'it': {(0, 0), (0, 3), (1, 2), (2,0)}
'what': {(0, 2), (1, 0)}.

4 Vyhledávání v metrických prostorech

Vyhledávání vždy bylo jedno z nejvýznamnějších dat zpracujících operací, nicméně získání přesné shody, což je typické pro tradiční databáze, není pro data v dnešní době ani proveditelné ani smysluplné. Důvodem je to, že stále se rozšiřující moderní digitální data postrádají strukturu a přesnost. Problém vyhledávání je obecně omezený typem dat uložených v databázi, metodou porovnání jednotlivých dat a specifikací dotazu, kterým uživatelé vyjadřují jejich potřeby informací. Zacházet s daty jako s metrickými objekty obecně přináší velkou výhodu, protože mnoho datových tříd a strategie hledající informace se přizpůsobí podle metrického pohledu, a proto se mohou aplikovat jednotlivé indexující techniky na specifické problémy vyhledávání. Schéma indexace, které dovoluje různé formy dotazů, nebo které může být rozšířeno dodatečnou funkčností, je důležitější než schéma lepší v určitých ohledech, avšak nemůže být rozšířeno.

Velmi užitečné vyhledávací paradigma je určit blízkost, podobnost nebo nepodobnost dotazu objektu vzhledem k objektům uložených v databázi. Objekty, které jsou blízko daného dotazu, tvoří množinu odpovědí. Problém vyhledávání může být obecně popsán jako v [35]:

Def. 2.1. Nechť D je doména, d míra vzdálenosti na D, a (D, d) metrický prostor. Je dána množina $X \subseteq D$ s n elementy nebo strukturami dat tak, že dotazy blízkosti jsou zodpovězeny efektivně.

Ačkoli již existuje několik typů podobnostních dotazů, další se očekávají v budoucnu. Základní typy jsou známé jako dotazy na *rozsah podobnosti* a *nejbližšího souseda(y)*.

Ve vzdálenostním prostoru, jediná možná operace na datových objektech je počítání vzdálenostní funkce na párech objektů, které splňují *trojúhelníkovou nerovnost*. Na rozdíl od toho, objekty v *koordinovaném prostoru* (koordinovaný prostor je zvláštní případ metrického prostoru) mohou být brány jako vektory. Takové prostory splňují některé další vlastnosti, které mohou být využity v návrzích (indexových) struktur uložení. Pro přehledy technik, které využívají vlastnosti koordinovaného prostoru se podívejme například na [4].

Hlavní důvody dle [35] proč brát v úvahu problém hledání vzdáleností dat vážně jsou následující:

1 Existují aplikace, kde kritéria blízkosti nenabízí žádné zvláštní vlastnosti, ale vzdálenosti ano, takže metrické hledání se stává jedinou volbou.
2 Mnoho specializovaných řešení pro hledání blízkosti nefunguje lépe než indexové techniky založené na vzdálenostech. Metrické hledání tak tvoří schopnou alternativu.

4.1 Metrický prostor

Hledání podobnosti může být bráno jako proces získávání datových objektů v pořadí jejich vzdáleností nebo nepodobnosti z daného dotazovacího objektu. To je druh *klasifikace* objektů s ohledem na dotazovací objekt, kde kritérium pozice je míra vzdálenosti. Ačkoli tento princip pracuje pro jakékoliv vzdálenostní míry, omezíme možnou množinu měr na **metrické požadavky.**

Předpokládejme metrický prostor M = (*D*, d) definovaný pro doménu objektů (nebo objektových klíčů nebo indexovaných rysů) *D* a celkovou (vzdálenostní) funkci *d.* V tomto metrickém prostoru vlastnosti funkce *d: D* x *D* **a** R, někdy označovány jako postuláty metrického prostoru, jsou typicky charakterizovány jako:

(p1) $\forall x, y \in D, d(x, y) \geq 0$ nezápornost,

(p2) $\forall x, y \in D, d(x, y) = d(y, x)$ symetrie,

(p3) $\forall x \in D, d(x, x) = 0$ reflexivita,

(p4) $\forall x, y \in D, x \neq y \Rightarrow d(x, y) > 0$ spolehlivost,

(p5) $\forall x, y, z \in D, d(x, z) \leq d(x, y) + d(y, z)$ trojúhelníková nerovnost.

Pro stručnost, někteří autoři nazývají *metrickou funkci* jednoduše *metrika*. Pokud vzdálenostní funkce nesplňuje vlastnost spolehlivosti (p4), je to nazýváno *klamná metrika*. Takové funkce mohou být transformovány do standardních metrik pomocí jakéhokoliv páru objektů s nulovou vzdáleností. Taková transformace je správná, pokud trojúhelníková nerovnost (p5) platí a lze prokázat $d(x, y) = 0 \Rightarrow \forall z \in D, d(x, z) = d(y, z)$.

Na druhé straně, pokud vlastnost souměrnosti (p2) neplatí, mluvíme o *kvazi metrice*, tzn. že funkce musí být asymetrická. Existují techniky k tomu, aby transformovaly asymetrické vzdálenosti do symetrické formy, např. $d_{sym}(x, y) = d_{asym}(x, y) + d_{asym}(y, x)$.

Jako poslední variantu funkce metrických vzdáleností uvádím tzv. *super metriku* nebo také *ultra metriku*, která splňuje následující trojúhelníkovou nerovnost:

$$\forall x, y, z \in D, d(x, z) \leq \max\{d(x, y), d(y, z)\}. \quad (4.1)$$

4.2 Míra vzdálenosti

Způsob počítání blízkosti objektů v dané doméně představují vzdálenostní funkce metrických prostorů. Vzdálenostní funkce jsou často uzpůsobeny ke konkrétnímu použití nebo třídě možných aplikací.

Míra vzdálenosti může být rozdělena na dvě skupiny v závislosti na povaze vrácených hodnot:

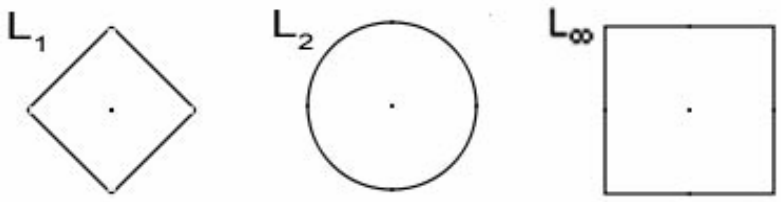
1. **diskrétní** - vzdálenostní funkce, které vracejí jen malou (předdeklarovanou) sadu hodnot. Příkladem je např. *editační vzdálenost.*
2. **spojitý** - vzdálenostní funkce, ve kterých mohutnost množin vrácených hodnot je velmi velká nebo nekonečná. Příkladem je třeba *Euklidovská vzdálenost* mezi vektory.

V následujícím textu provedeme hlavně přehled metrických funkcí užívaných pro komplexní typy dat jako mnohorozměrné vektory, řetězce nebo množiny [35].

4.2.1 Editační vzdálenost

Blízkost sekvencí symbolů(řetězců) může být účinně měřeno *editační vzdáleností*, jinak zvanou *Levenshteinovou vzdáleností*, představenou v [20]. Vzdálenost mezi dvěma řetězci $x = x_1 \ldots x_n$ a $y = y_1 \ldots y_m$ je definována jako minimální množství atomických editačních operací (vložit, smazat a vyměnit) potřebných k tomu, aby transformovaly řetězec *x* na řetězec *y*. Atomické operace jsou definované formálně tímto způsobem:

- **vložit** znak c do řetězce x na pozici i : $ins(x, i, c) = x_1x_2 \ldots x_icx_{i+1} \ldots x_n$,
- **vymazat** znak z pozice i z řetězce x: $del(x, i) = x_1x_2 \ldots x_{i-1}x_{i+1} \ldots x_n$,
- **vyměnit** znak na pozici *i* v *x* s novým znakem *c*: $replace(x, i, c) = x_1x_2 \ldots x_{i-1}cx_{i+1} \ldots x_n$.



Obr 4.1 Množiny bodů ve stejné vzdálenosti od centrálního bodu pro různé L_p metriky.

Zobecněná editační vzdálenostní funkce přiřazuje váhy (pozitivní reálná čísla) jednotlivým atomickým operacím. Z toho důvodu, vzdálenost mezi řetězci x a y je minimální hodnota sumy vážených atomických operací potřebných k transformaci x na y. Editační vzdálenost není symetrická (je porušena vlastnost (p2) definovaná výše), jestliže váhy vložené a smazané operace se liší, a proto není metrickou funkcí.

4.2.2 Stromová editační vzdálenost

Tato vzdálenost je známá mírou blízkosti pro stromy, značně studované v [1]. Stromová editační vzdálenostní funkce definuje vzdálenost mezi dvěmi stromovými strukturami jako minimální cenu potřebnou k přeměně zdrojového stromu na cílový strom využívající předdeklarované sady stromových uspořádaných operací, jako např. vložení nebo zrušení uzlu. Jednotlivé ceny editačních atomických operací mohou být konstantní pro celý strom, nebo mohou kolísat s úrovní ve stromu, ve kterém je operace uskutečněna. Různé váhy u úrovní ve stromu jsou proto, že vložení jednotlivých uzlů blízko kořene může být více významné než přidání nového koncového uzlu. Stromová editační vzdálenost může být také využita pro míru strukturální nepodobnosti XML dokumentů [9].

4.2.3 Minkowského vzdálenost

Minkowského vzdálenostní funkce tvoří celou rodinu metrických funkcí určených jako L_p metriky, protože jednotlivé případy závisí na číselných parametrech p. Tyto funkce jsou definované na n-rozměrných vektorech reálných čísel jako:

$$L_p[(x_1,...,x_n),(y_1,...,y_n)] = \sqrt[p]{\sum_{i=1}^{n} | x_i - y_i |^p} \quad , \tag{4.2}$$

kde L_1 metrika je známá jako *Manhattanská vzdálenost*, L_2 vzdálenost značí známou Euklidovskou vzdálenost, a $L_\infty = \max_{i=1}^{n} | x_i - y_i |$ je nazývána *Chebyshevova* neboli *šachovnicová vzdálenost*.

Manhattanská vzdálenost někdy také zvaná newyorská metrika, byla inspirována pravoúhlým systémem ulic na Manhattanu. V dvourozměrném pozorování, jde o vzdálenost dvou bodů v rovině měřenou po odvěsnách pravoúhlého trojúhelníku, zatímco euklidovská vzdálenost je měřená po přeponě.

Euklidovská vzdálenost je z rodiny L_p metrik nejpoužívanější. V případě, že by objekty byly charakterizovány pouze dvěma znaky, pak je lze zakreslit jako bod v rovině a Euklidovská vzdálenost je definována jako délka úsečky spojující příslušné dva body. Tato vzdálenost je vhodná k použití v situacích, kdy měřené znaky jsou nezávislé, normálně rozdělené se stejnými rozptyly a v případech, kdy znaky jsou věcně podobné a stejně důležité pro klasifikaci dat.

4.2.4 Mahalanobisova vzdálenost

Mahalanobisova vzdálenost je založena na korelaci mezi proměnnými [44]. Je to užitečný způsob určení podobnosti neznámého vzorku dat vzhledem ke známému vzorku. Odlišuje se od euklidovské vzdálenosti tak, že bere v úvahu korelaci datové sady. Tato vzdálenost může být definována jako rozdílnost mezi dvěma náhodnými vektory *x* a *y* se stejným rozdělením a s kovarianční maticí S:

$$d(x,y) = \sqrt{(x-y)^T S^{-1}(x-y)}\,. \tag{4.3}$$

Pokud kovarianční matice je matice identity, Mahalanobisova vzdálenost se redukuje na Euklidovskou vzdálenost. Pokud však kovarianční matice je diagonální, výsledná vzdálenost se nazývá *normalizovaná Euklidovská vzdálenost:*

$$d(x,y) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{s_i^2}}\,, \tag{4.4}$$

kde σ_i je směrodatná odchylka x_i na vzorové sadě.

4.2.5 Jaccardův koeficient

Zde si představíme míru podobnosti, která je vhodná pro množiny. Předpokládejme dvě množiny A a B, *Jaccardův koeficient* je definovaný jako

$$d(A,B) = 1 - \frac{|A \cap B|}{|A \cup B|}\,. \tag{4.5}$$

Tato vzdálenostní funkce je jednoduše založená na koeficientu mezi mohutností průniku a sjednocení srovnávaných množin. Jako příklad aplikace, která pracuje s množinami, předpokládejme přístup k log souboru webových adres (URL) zpřístupněných návštěvníky v internetové kavárně. Spolu s adresami jsou v logu uloženy také návštěvnické identifikace. Chování uživatele prohledávající Internet může být vyjádřeno množinou navštívených stránek a Jaccardův koeficient může být použit k určení podobnosti (nebo nepodobnosti) jednotlivých zájmů uživatelova hledání.

4.2.6 Hausdorffova vzdálenost

Ještě komplikovanější vzdálenostní míra definovaná na množinách je *Hausdorffova vzdálenost* [15]. Na rozdíl od Jaccardova koeficientu, kde jakékoliv dva prvky množin musí být buď shodné nebo úplně odlišné, Hausdorffova vzdálenost porovnává elementy pomocí vzdálenostní funkce d_e. Hausdorffova vzdálenost je definovaná tímto způsobem. Předpokládejme:

$$d_p(x,B) = \inf_{y \in B} d_e(x,y)\,, \tag{4.6}$$

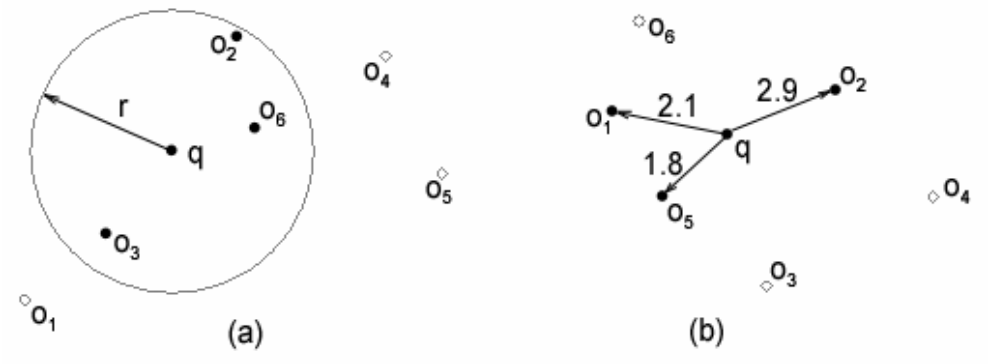
$$d_p(A,y) = \inf_{x \in A} d_e(x,y)\,, \tag{4.7}$$

$$d_s(A,B) = \sup_{x \in A} d_p(x,B)\,, \tag{4.8}$$

$$d_s(B,A) = \sup_{y \in B} d_p(A,y)\,. \tag{4.9}$$

Potom Hausdorffova vzdálenost nad množinami A, B je: $d(A, B) = max\{d_s(A, B), d_s(B, A)\}$.

Stručně řečeno, dvě množiny jsou v Hausdorffově vzdálenosti *r* od sebe právě tehdy, když nějaký bod jedné množiny je ve vzdálenosti *r* z nějakého bodu jiné množiny. Typická aplikace je porovnání tvarů ve zpracování obrazu, kde každý tvar je definován množinou bodů v 2-rozměrném prostoru.



Obr 4.2 (a) Rozsahový dotaz, (b) Dotaz na nejbližšího souseda 3NN(q).

4.2.7 Časová složitost

Počítání vzdálenosti je obecně netriviální proces, který jistě bude mnohem víc výpočetně intenzivní, než porovnávání s klíčovým heslem jako se využívá v tradičních vyhledávacích strukturách. Například, L_p normy jsou vypočítány v lineárním čase závislém na rozměrnosti n prostoru, vzdálenost kvadratické formy je mnohem dražší, protože to zahrnuje násobení maticí M.

Vysoká výpočetní složitost metrických vzdálenostních funkcí nutí metrické indexové struktury k důležitému cíli a to zejména minimalizování počtu vzdálenostních vyhodnocení.

4.3 Podobnostní dotazování

Dotaz na podobnost je definovaný explicitně nebo implicitně dotazovacím objektem q a je požadováno omezení na podobu a rozsah blízkosti, typicky vyjádřeno jako vzdálenost. Odpověď na dotaz vrací všechny objekty, které splňují výběr podmínek, tedy předpokládáme objekty blízko u daného dotazovacího objektu. Toto jsou základní druhy podobnostních dotazů, více v [35]:

4.3.1 Rozsahový dotaz

Pravděpodobně nejběžnější typ dotazu podobnosti je *rozsahový dotaz podobnosti - R(q, r).* Dotaz je specifikovaný dotazovacím objektem $q \in D$, s dotazovacím poloměrem r jako vzdálenostní omezení. Dotaz znovu získá všechny objekty nalezené ve vzdálenosti r od q, formálně:

$$R(q,r) = \{o \in X, d(o,q) \leq r\}. \tag{4.10}$$

Jak vidíme, dotazovací objekt q se nemusí vyskytovat v kolekci $X \subseteq D$ a jediné omezení na q je, že náleží do metrické domény D. Pokud vyhledávací poloměr je nula - R(q, 0), dotaz je nazýván *bodový dotaz* neboli *přesná shoda*. V tomto případě hledáme identickou kopii (nebo kopie) dotazovacího objektu q. Nejobvyklejší použití tohoto typu dotazu je v algoritmech mazání, kdy chceme lokalizovat objekt k odstranění z databáze.

Někdy může být obtížné specifikovat poloměr dotazu bez určitých znalostí dat a vzdálenostní funkce, což je u tohoto typu dotazu nutné. Jestli je specifikován příliš malý dotazovací poloměr, může se stát, že bude vrácena prázdná množina. Na druhé straně, pokud dotazovací poloměr je příliš velký, dotaz může být výpočetně drahý a množiny odpovědí obsahují četné nevýznamné objekty.

4.3.2 Dotaz na nejbližšího souseda

Alternativní způsob jak hledat podobné objekty je použít *dotazy na nejbližšího souseda*. Tento dotaz nalézá nejbližší objekt k danému dotazovanému objektu, to jest nejbližšího souseda q. Pojem může být zobecněný do případu, kde hledáme k nejbližších sousedů, tedy kNN(q). Množina odpovědí může být definována tímto způsobem:

$$kNN(q) = \{R \subseteq X, |R| = k \wedge \forall x \in R, y \in X - R : d(q,x) \leq d(q,y)\}. \quad (4.11)$$

Když několik objektů leží ve stejné vzdálenosti od k-tého nejbližšího souseda, vazby jsou vyřešeny libovolně.

4.3.3 Reverzní dotaz na nejbližšího souseda

Často je dobré vědět, které objekty vidí dotazovací objekt q jako svého nejbližšího souseda. Toto je známé jako *reverzní vyhledávání nejbližšího souseda*, obyčejně označováno *kRNN(q)*.

Poslední práce, jako [32], zvýraznili význam obrácených dotazů na nejbližšího souseda v rozhodovacích podpůrných systémech, uskladnění dokumentů, a managementu mobilních zařízení. Množina odpovědí *kRNN(q)* dotazu může být definovaná následovně:

$$kRNN(q) = \begin{Bmatrix} R \subseteq X, \forall x \in R : q \in kNN(x) \wedge \\ \forall x \in X - R : q \notin kNN(x) \end{Bmatrix}. \quad (4.12)$$

Zde vidíme, že dokonce objekt lokalizovaný daleko od dotazovacího objektu q může patřit kRNN(q) množině odpovědí. Zároveň objekt blízko q nemusí nutně být členem kRNN(q) výsledku.

4.3.4 Podobnostní spojení

Podobnostní spojení mezi dvěma množinami $X \subseteq D$ a $Y \subseteq D$ získávají všechny páry objektů $(x \in X, y \in Y)$ jejichž vzdálenost nepřesáhla daný práh $\mu \geq 0$. Výsledek podobnostního spojení J(X, Y, μ) je definován jako: $J(X,Y,m) = \{(x,y) \in X \times Y : d(x,y) \leq m\}$.

Pokud práh μ = 0, dostaneme tradiční *přirozené spojení* a pokud se množiny X a Y shodují, tj. X = Y, mluvíme o *podobnostním samo-spojení* a značí se SJ(μ) = J(X, X, μ), kde X je hledaná množina.

4.3.5 Kombinace dotazů

Jako rozšíření dotazů, můžeme definovat další druhy dotazů jako kombinaci předchozích uvedených. Lze spojit např. rozsahový dotaz s dotazem na nejbližšího souseda tak, abychom dostali kNN(q, r) s odpovědí stanovenou:

$$kNN(q,r) = \begin{Bmatrix} R \subseteq X, |R| \leq k \wedge \forall x \in R, y \in X - R : \\ d(q,x) \leq d(q,y) \wedge d(q,x) \leq r \end{Bmatrix}. \quad (4.13)$$

Výsledek máme omezený ze dvou stran, nejprve všechny objekty ve výsledkové množině by měly ležet ve vzdálenosti ne větší než r a pokud existuje víc než k z nich, jen prvních (tj. nejbližších) k je vráceno.

4.3.6 Komplexní podobnostní dotazy

Účinné zpracování dotazů sestávajících se z více než jednoho podobnostního predikátu, tj. *komplexní dotazy na podobnost* se podstatně liší od tradičního (Booleova) zpracování dotazu. K porozumění problému zvažujme dotaz na kruhové tvary červené barvy. K tomu, abychom našli nejlepší shodu,

nestačí získat nejlepší shody pro rysy barvy a tvary. Přirozeně nejlepší shoda pro celý dotaz nemusí být nejlepší shodou pro jednotlivé (barevné nebo tvarové) predikáty.

Pro tento účel, [Fagin, 1996] navrhl tzv. A_0 algoritmus. Tento algoritmus předpokládá, že pro každý dotazovací predikát máme indexovou strukturu schopnou vrátit objekty klesající podobnosti. Pro každý predikát *i*, algoritmus postupně vytváří množinu X_i obsahující objekty, které se nejvíce shodují k dotazovému predikátu. Tato stavební fáze pokračuje až dokud všechny množiny X_i neobsahují přinejmenším *k* běžných objektů. Pro všechny objekty $o \in \bigcup_i X_i$, algoritmus ohodnocuje veškeré dotazovací predikáty a stanovuje jejich finální pozice. Pak prvních *k* objektů je vráceno jako výsledek. Tento algoritmus je správný, ale jeho výkon není moc optimální a očekávané ceny provedení dotazu mohou být dosti vysoké.

V [7] se soustředí na komplexní dotazy podobnosti vyjádřeny skrz generický jazyk. Předpokládají, že dotazovací predikáty jsou z jediné domény, tj. ze stejného metrického prostoru. Navrhovaný proces vyhodnocení je založený na vzdálenostech mezi hodnotami rysů, protože metrické indexy mohou užívat jen vzdálenosti k tomu, aby ohodnotily predikáty. Řešení navrhuje, že index by měl zpracovat komplexní dotazy jako celek. Možnost realizovat takový přístup je demonstrovaný skrz rozšířený M-strom. Zkušební výsledky ukazují, že výkon rozšířeným M–stromem [8] je konzistentně lepší než A_0 algoritmus. Rozšířený M-strom, je schopen porovnat různé rysy s libovolnými vzdálenostními funkcemi.

Algebra podobnosti s váhami byla představena v [6]. To je zevšeobecnění relační algebry k tomu, aby dovolila formulaci komplexních dotazů na podobnost nad multimediálními databázemi. Hlavní příspěvek této práce je, že kombinuje uvnitř jednotlivých rámců několik významných aspektů podobnostního vyhledávání, jako jsou nové operátory (Top a Cut), váhy k vyjádření uživatelské přednosti a skóre k ohodnocení výsledků hledání.

4.4 Základní principy dělení

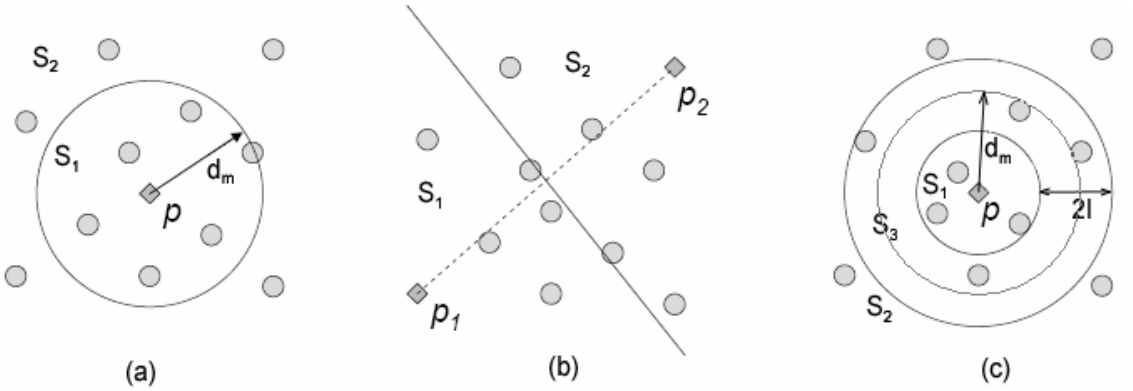
K efektivnímu vyhledávání dat a manipulaci s nimi je nutné vhodně množinu strukturovat. Dělení je obecně jedním z nejzákladnějších principů jakékoliv struktury uložení, zaměřující se na dělení vyhledávacího prostoru tak, že jsou při dotazu prohledávány pouze některé části prostoru. Je dána množina $S \subseteq D$ objektů v metrickém prostoru M = (*D*, d). V následujícím textu, krátce charakterizujeme některé techniky dělení této množiny, jak je popsáno v [35].

4.4.1 Sférické dělení

Toto dělení rozděluje množinu S do podmnožin S_1 a S_2 s využitím sférického řezu s ohledem na $p \in D$, kde p je *pivot*, vybraný libovolně. Nechť d_m je střední hodnota množiny $\{d(o_i, p), \forall o_i \in S\}$. Pak všechny $o_j \in S$ jsou rozděleny do S_1 nebo S_2 podle následujících pravidel:

- $S_1 = \{o_j \mid d(o_j, p) \leq d_m\}$, (4.14)
- $S_2 = \{o_j \mid d(o_j, p) \geq d_m\}$. (4.15)

Pokud střední hodnota není jedinečná, redundantní podmínky ≤ a ≥ zaručí rovnováhu. Toto je provedeno přidělením každého prvku ve vzdálenosti střední hodnoty jedné z podmnožin libovolně, ale vyváženě.



Obr 4.3 Příklady dělení: (a) sférické dělení, (b) obecné dělení na poloroviny a (c) dělení vyjmutím středu.

4.4.2 Obecné dělení na poloroviny

Toto dělení může být považované za ortogonální podobu sférického dělení, takže také rozděluje množinu S do podmnožin S_1 a S_2 a to podle dvou referenčních náhodně vybraných pivotů $p_1, p_2 \in D$. Všechny další objekty jsou přiřazené množině S_1 nebo S_2 v závislosti na jejich vzdálenostech od vybraných pivotů tímto způsobem:

- $S_1 = \left\{ o_j \middle| d(p_1, o_j) \leq d(p_2, o_j) \right\}$, (4.16)
- $S_2 = \left\{ o_j \middle| d(p_1, o_j) \geq d(p_2, o_j) \right\}$. (4.17)

Na rozdíl od sférického dělení, zobecněná hyper-rovina negarantuje vyvážené rozdělení.

4.4.3 Dělení vyjmutím středu

Toto dělení s vyjmutím středu je modifikace sférického dělení. Rozdíl je v tom, že množina S se nerozděluje na dvě, ale na tři disjunktní podmnožiny S_1, S_2 a S_3. Hlavní představa dělení vyjmutím středu je vynechat body blízko prahu d_m při definování podmnožin S_1 a S_2, kde vyjmuté body tvoří třetí podmnožinu S_3. S takovým uspořádáním, hledání podobných objektů vždy ignoruje alespoň jednu z podmnožin S_1 nebo S_2, pod podmínkou, že vyhledávací selektivita je menší než *tloušťka* zóny vyloučení. Vyjmuté body nemohou být samozřejmě ztraceny, takže mohou být buď považovány za body tvořící třetí podmnožinu, nebo pokud je množina velká, základ nového rozdělovacího procesu. Objekty jsou do podmnožin přiřazeny následovně::

- $S_1 = \left\{ o_j \middle| d(o_j, p) \leq d_m - l \right\}$, (4.18)
- $S_2 = \left\{ o_j \middle| d(o_j, p) > d_m + l \right\}$, (4.19)
- $S_3 = ostatní$.

4.4.4 Rozšíření

Základní rozdělující principy mohou být rozšířeny několika způsoby a to přidáním dalších prahů, čímž se z binárního dělení stává vícenásobné dělení, a nebo pokud dělení probíhá rekurzivně, lze vybudovat stromovou strukturu, která je prakticky využitelná při návrhu indexových struktur.

4.5 Principy provedení dotazu

V následujícím textu diskutujeme některé obecné, dosti abstraktní principy provádění dotazů na podobnost. Kromě toho taky věnujeme pozornost rozdělovacím principům, strategiím pro provedení dotazu formující další důležitou část některých vyhledávacích struktur, protože mohou významně ovlivňovat efektivitu odpovídání dotazů.

Pro demonstraci obecných principů vyhledávacích strategií použitých v indexových strukturách [35], předpokládáme způsob hypotetické organizace metrických dat. Tedy vstup N = (G,R(G)) struktury skládající se z množiny G metrických objektů nebo jiných položek a specifikace *hraničícího regionu* R(G). Tento region představuje omezení na metriku, která musí být splněna všemi elementy $o \in G$.

V praxi mohou být hraničící regiony vytvořeny větším počtem komplexních podmínek. Každý prvek zpravidla patří právě jedné skupině (jedné množině G) zatímco jednotlivé hraničící regiony se mohou překrývat. Pro jednoduchost předpokládejme, že položky tvoří hierarchii a hledání vždy začíná v kořenovém vstupu. Jakmile kterýkoliv podobnostní dotaz *q* vrátí množinu objektů, můžeme vždy definovat hraničící region kolem objektů analogicky. Takový region označíme R(q) .

4.5.1 Přírůstkové hledání podobnosti

Toto hledání může poskytovat objekty v pořadí klesající podobnosti bez předem explicitně specifikovaného počtu nejbližších sousedů. Toto je zvláště důležité v interaktivních databázových aplikacích. Přírůstkový aspekt také poskytuje významné výhody v situacích, kde počet požadovaných sousedů je předem neznámý, například když jsou zpracovány komplexní dotazy podobnosti.

Přírůstkový algoritmus nejbližšího souseda navrhovaný v [13] je použitelný kdykoliv, když vyhledávací prostor je strukturovaný hierarchickým způsobem. Hlavní rozdíl leží v definování specializované vzdálenostní funkce na místo pokrývajících regionů, které dovolují přímější vysvětlení přírůstkového algoritmu vyhledávání.

4.6 Vyhýbání se výpočtu vzdáleností

Vzhledem k často velmi drahým výpočtům vzdáleností je dosti důležité omezit počet vzdálenostních výpočtů jak jen je to možné. Pro tento účel musí být aplikovány nejen podmínky prořezávání, abychom se vyhnuli zpřístupnění irelevantních množin objektů, ale také minimalizovat počet vzdálenostních výpočtů. Odůvodnění takových strategií je použít již ohodnocené vzdálenosti mezi nějakými objekty, k určení hranic na vzdálenostech mezi dalšími objekty a to při vhodném použití předpokladů metrického prostoru, tj. trojúhelníková nerovnost, souměrnost, a nezápornost.

Některé *hraničící strategie*, původně navržených v [14], představují hlavní pravidla prořezávání, která jsou použita ve zvláštním tvaru prakticky ve všech indexových strukturách pro metrické prostory.

4.6.1 Vzdálenostní omezení Objekt – Pivot

Základním typem hraničícího omezení je *vzdálenostní omezení objekt-pivot*, které je obvykle aplikováno na koncový uzel, obsahující data tj. metrické objekty hledané sbírky.

Příklad takového omezení je např. v [35]. Formálně je však popsané v Lemma 1.1.

LEMMA 1.1 Je dán metrický prostor M = (D , d) a tři libovolné objekty q, p, o ∈ D, je vždy zaručeno

$$|d(q,p) - d(p,o)| \leq d(q,o) \leq d(q,p) + d(p,o) . \quad (4.20)$$

Následkem toho může být vzdálenost d(q, o) ohraničena zdola i shora a vzdálenosti d(q, p) a d(p, o) jsou známy.

4.6.2 Vzdálenostní omezení Rozsah – Pivot

Výše popsané omezení předpokládá, že všechny vzdálenosti mezi databázovými objekty o_i a příslušným pivotem p jsou známy. Nicméně některé metrické struktury se pokouší minimalizovat prostor potřebný k vytvoření indexu, takže ukládání takového množství dat není přijatelné. Alternativa je skladovat jen rozsah (vzdálenostní interval), ve kterém se databázové objekty vyskytují s ohledem na p. Zde můžeme použít slabší podmínku zvanou *vzdálenostní omezení rozsah - pivot.*

Pro užitečnost tohoto omezení vezměme v úvahu rozsahové dotazy. Pokud spodní hranice je větší než dotazovací poloměr r, jsme si jisti, že žádný oprávněný objekt nemůže být nalezen a uzel nemusí být zpřístupněn. Na druhé straně, jestliže horní hranice je menší nebo rovna r, pak můžeme rozhodnout, že všechny objekty se kvalifikují a přímo zahrnují veškeré potomky objektů v dotazovací množině odpovědí. Toto omezení lze formovat takto:

LEMMA 1.2 Máme dán metrický prostor M = (D , d) a objekty o,p ∈ D takové, že $r_l \le d(p,o) \le r_h$ *, je dán objekt* $q \in D$ *a přidružená vzdálenost d(q, p). Vzdálenost d(q, o) může být omezena rozsahem:*

$$\max\{d(q,p) - r_h, r_l - d(q,p), 0\} \le d(q,o) \le d(q,p) + r_h, \tag{4.21}$$

kde r_l a r_h jsou krajní body intervalu $[r_l, r_h]$, ve kterém se vyskytuje vzdálenost od pivota p k objektu o.

4.6.3 Vzdálenostní omezení Pivot – Pivot

Předešlé dva principy vedou k posílení výkonu v algoritmech vyhledávání. *Vzdálenostní omezení pivot - pivot* i přesto, že je slabší než předchozí dva, poskytuje určité výhody. Detailnější popis s příkladem naleznete v [35] a zde uvedeme pouze formalizaci tohoto principu:

LEMMA 1.3 Je dán metrický prostor M = (D , d) a objekty o,p, q ∈ D takové, že $r_l \le d(p,o) \le r_h$ *a* $r_l' \le d(q,p) \le r_h'$*. Vzdálenost d(q,o) může být ohraničena rozsahem:*

$$\max\{r_l' - r_h, r_l - r_h', 0\} \le d(q,o) \le r_h + r_h'. \tag{4.22}$$

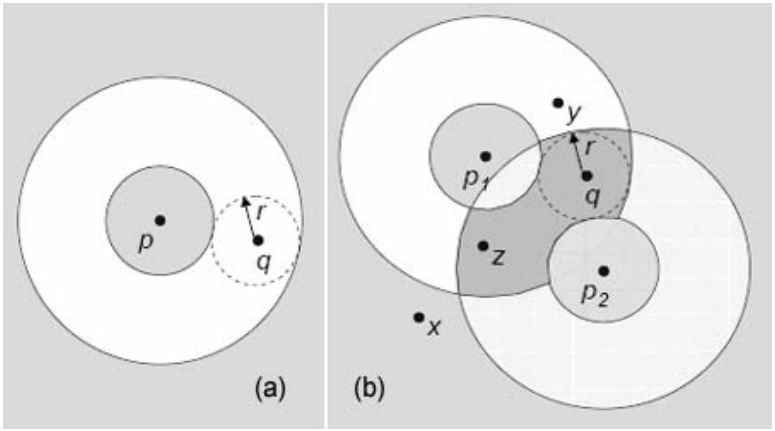
4.6.4 Vzdálenostní omezení Double – Pivot

Všechny tři předchozí přístupy využívají k urychlení procesu získání v metrických strukturách jediného pivota, v souladu s kruhovým rozdělovacím vzorem. *Vzdálenostní omezení double – pivot* je založeno na zobecněném rozdělování hyper-rovinou, tedy využívá dva pivoty k rozdělení metrického prostoru. Stejně jako u předchozí strategie odkazuji na [35] pro lepší pochopení. Formální definice tohoto omezení je dána v Lemma 1.4.

LEMMA 1.4 Předpokládejme metrický prostor M = (D , d) a objekty o, $p_1, p_2 \in D$ *takové, že* $d(o,p_1) \le d(o,p_2)$*. Je dán dotazovací objekt* $q \in D$ *a vzdálenosti* $d(q,p_1)$ *a* $d(q,p_2)$*, vzdálenost d(q, o) je zespodu ohraničena tímto způsobem :*

$$\max\left\{\frac{d(q,p_1) - d(q,p_2)}{2}, 0\right\} \le d(q,o). \tag{4.23}$$

Důležitá informace u této strategie je ta, že nepoužívá žádné již ohodnocené vzdálenosti od pivota k databázovému objektu. Pokud známe takové vzdálenosti pro oba pivoty, měli bychom jednoduše použít dvakrát Lemma 1.1, pro každý pivot odděleně.



Obr 4.4 Příklady filtrovacích technik: (a) využívající jediného pivota, (b) využívající kombinaci pivotů.

4.6.5 Filtrování

Pro dosažení větší míry prořezávání může být zkombinováno několik pivotů do jednotlivých *filtrovacích* technik [Dohnal, 2004]. Výchozí idea je ukázána na obr. 4.4. Tomuto konceptu je dána pevná forma v následujícím lemma.

LEMMA 1.5 Předpokládejme metrický prostor M=(D , d) a množinu pivotů P={p_1,…, p_n}. Definujme mapovací funkci $y : (D,d) \to (R^n, L_\infty)$ *následovně:*

$$y(o) = (d(o,p_1), d(o,p_2), ..., d(o,p_n)). \tag{4.24}$$

Poté můžeme ohraničit vzdálenost d(q, o) zdola:

$$L_\infty(y(q), y(o)) \le d(q,o). \tag{4.25}$$

Zobrazení $y(\cdot)$ vrací vektor vzdáleností z objektu *o* všem pivotům v *P*. Pro databázový objekt, vektor ve skutečnosti obsahuje předběžné vzdálenosti k pivotům. Na druhé straně, použití $y(\cdot)$ na dotazovací objekt q vyžaduje počítání vzdáleností od dotazovacího objektu ke všem pivotům z *P*.

4.7 Transformace metrického prostoru

Transformováním jednoho metrického prostoru do jiného ve skutečnosti znamená zobrazení všech objektů do nové domény, takže může být používána odlišná vzdálenostní funkce. Obecně měníme oboje, jak objekty tak i metrickou funkci, ale některé aplikace mohou požadovat pouze změnu metrické funkce nebo změnu domény. Nová doména dovoluje dotaz na podobnost v transformovaném prostoru proto, aby byl nahrazen dotazem v originálním metrickém prostoru. Motivací může být možnost méně nákladného počítání, a v tomto případě mluvíme o *vloženém metrickém prostoru*.

4.7.1 Metrické hierarchie

Nejprve definujme transformaci metrického prostoru $M_1 = (D_1, d_1)$ do metrického prostoru $M_2 = (D_2, d_2)$ jako funkci $f : D_1 \to D_2$ takovou, že

$$\forall o_1, o_2 \in D_1 : d_1(o_1, o_2) \approx d_2(f(o_1), f(o_2)). \tag{4.26}$$

Všimněme si, že vzdálenost v transformovaném metrickém prostoru se nemusí přesně rovnat originální vzdálenosti. Pro účely podobnostního vyhledávání je nezbytné definovat vztah mezi vzdálenostmi v originálních a transformovaných metrických prostorech.

Řekněme, že d_1 je *spodně-hraničící vzdálenostní* funkce d_2 :

$$\forall o_1, o_2 \in D_1 : d_1(o_1, o_2) \leq d_2(f(o_1), f(o_2)) . \quad (4.27)$$

Předpokládejme, že d_1 není spodně-hraničící vzdálenostní funkce d_2. Pak můžeme definovat novou *vzdálenostní funkci* $d_{1s}(o_1,o_2) = s_{d_1 \to d_2} . d_1(o_1,o_2)$ takovou, že d_{1s} je spodně-hraničící vzdálenostní funkce d_2, kde $0 < s_{d_1 \to d_2} < 1$ je reálné číslo zvané *měřítko*. Zřetelně není vždy možné definovat d_{1s}, protože vhodné měřítko nelze získat pro každý pár vzdálenostních funkcí d_1, d_2. Nicméně pokud měřítko $s_{d_1 \to d_2}$ existuje, pak lze nalézt nekonečné množství dalších hodnot měřítka. Maximální hodnota měřítka se nazývá *optimální měřítko*. Pro kompletní vysvětlení v [27].

Příklady nejvíce běžných spodně-hraničících vzdálenostních funkcí jako jsou L_p normy či vzdálenostní funkce kvadratické formy naleznete např. v [35].

4.7.2 Uživatelem definované metrické funkce

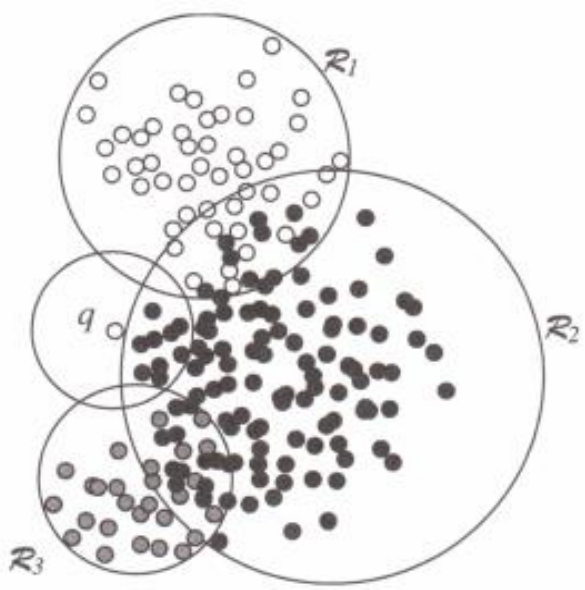
Uživatelé mají často různé představy o podobnosti, která nakonec určuje vyhodnocení a řazení databázových objektů. V dnešní době existují aplikace s velice komplexními doménami, jako multimédia nebo systémy dolování dat, kde jsou preference jednoduše za představivostí obyčejného uživatele. Nechť máme indexovou strukturu v metrickém prostoru, postavenou používáním metrické funkce d_b definované na doméně D. Další metrická funkce d_u na stejné doméně D je buď explicitně specifikovaná uživatelem nebo automaticky odhadnuta systémem. Aktuální výsledky uživatelského dotazu musí být vypočítány v souladu s d_u.

4.8 Přibližné hledání podobnosti

Hledání podobnosti v metrických prostorech je obecně drahé a nejmodernější metody stále neposkytují přijatelný čas odpovědi pro vysoce interaktivní aplikace, avšak v mnoha aplikacích je to dostačující k tomu, aby vykonávaly *přibližné podobnostní vyhledávání*. U tohoto typu vyhledávání je získána nepřesná výsledková množina a přitažlivost tohoto přístupu spočívá v tom, že přibližné hledání je typicky vykonané mnohem rychleji.

Techniky přibližného vyhledávání podobnosti nabízí velmi zlepšenou efektivitu za cenu určité nepřesnosti ve výsledcích. Hlavní představa přibližných algoritmů je zmírnit některá omezení na precizní hledání podobnosti, aby redukovaly vyhledávací náklady a nebo počet vzdálenostních výpočtů. Důvodem použití přibližného hledání podobnosti je to, že podoba mezi objekty je často subjektivní, tudíž je velmi obtížné vyjádřit jedinečnou přesnou funkcí. Uveďme např. obrazovou databázi, kde různí uživatelé by mohli určit různé výsledky jako podobné při zadání stejného dotazu. Intuitivní představa je však formálně definována matematickým vzorcem, tudíž subjektivnost není brána v úvahu. Proto by mohla být řízená nepřesnost s následkem rychlejšího hledání uživatelem tolerována. Dalším důvodem je to, že procesy vyhledávání podobnosti jsou iterativní, a tedy uživatel může začít hledání použitím počátečního obrazu a pokud není spokojen s výsledkem, může zadat další dotaz s použitím jednoho z vrácených obrazů.

Přístupy k přibližnému hledání podobnosti lze rozdělit na [10]:



Obr 4.5 Strategie zeslabující větvení může rozhodnout znepřístupnit region R_1 a R_3, které nesdílí žádné objekty s dotazovým regionem, i když překrývají dotazový region [35].

1. *Přístupy, které využívají transformace metrického prostoru*. Přibližnost je dosažena změnou objektové reprezentace a nebo vzdálenostní funkce s cílem snižování vyhledávacích nákladů. Techniky transformace jsou popsány v sekci 2.7. Pokud není aplikován filtrující krok, algoritmus vyhledávání je přibližný, tzn. je rychlejší za cenu falešných tref ve výsledku. Hlavní využití je ve vektorových prostorech.

2. *Přístupy, které redukují podmnožinu dat*. Tyto techniky se zaměřují na zlepšení výkonu zpřístupněním a analyzováním méně dat, než je technicky potřeba. Existují dvě základní strategie přibližnosti, které se používají k redukci dat:

    - *Strategie brzkého ukončení* zastaví algoritmus vyhledávání před jeho přirozeným koncem. Algoritmy podobnostního vyhledávání jsou iterativní procesy, ve kterých aktuální výsledková množina může být zlepšena v každém kroku. Precizní algoritmus se zastaví, když zjistí, že žádná další zlepšení nejsou možná. Na druhou stranu, přibližné algoritmy využívají podmínku zastavení k rozhodnutí brzkého ukončení algoritmu. Algoritmus se ukončí, když zjistí, že je malá šance na získání významně lepších výsledků.
    - *Strategie zeslabující větvení* – se vyhýbají zpřístupnění datových regionů, které nejsou pravděpodobné, že obsahující objekty patřící do výsledkové množiny. Precizní algoritmy podobnostního vyhledávání zpřístupní všechny datové oblasti překrývající dotazovací region a ostatní vyřadí. Strategie zeslabující větvení jsou založeny na definici přibližné *podmínky prořezávání*, která rozhoduje o odmítnutí regionů překrývajících dotazovací region. Datové oblasti jsou vyřazeny, když podmínky zjistí nízkou pravděpodobnost, že objekty se vyskytnou v prostoru sdíleném s dotazovacím regionem. Tyto strategie jsou zvláště užitečné pro metody přístupu založených na hierarchickém rozkládání prostoru.

4.8.1 Generické algoritmy

Algoritmy pro přibližné hledání podobnosti, které využívají brzké ukončení a strategie zeslabující větvení mohou být snadno získány modifikováním generických algoritmů podobnostního vyhledávání, které jsou popsány v sekci 2.5. Jediný rozdíl od těchto algoritmů je, že test překrytí pro regiony je nahrazený čistící podmínkou *Prune* a *Stop* podmínka je využita pro rozhodnutí předčasného ukončení.

Obecná podmínka zastavení *Stop*(response, x_s), bere jako argumenty aktuální výsledkovou množinu *response* (množina oprávněných objektů nalezených až po aktuální iteraci) a parametr přibližnosti x_s a vrací *true*, pokud strategie ukončení určí, že požadavky přibližnosti byly splněny. Argument *response* přechází do podmínky zastavení, aby zdůraznil možnost definování strategií, které analyzují aktuální množinu odpovědí k tomu, aby odhadovaly kvalitu aktuální přibližnosti.

Obecná podmínka prořezávání *Prune(R(G), R(q), x_p)* bere jako argumenty dotazovací území R(q), hraničící území R(G) a parametr přibližnosti x_p a vrací *true*, pokud strategie prořezávání určí, že vstup, pokrytý datovou oblastí může být vyřazen podle parametru přibližnosti x_p. Je důležité zdůraznit, že region R(G) je získaný bez zpřístupnění vstupu. Parametry přibližnosti x_s a x_p jsou využívány k přizpůsobení dohody mezi efektivitou a přesností. Hodnoty korespondující s vysokým výkonem nabízejí nízkou přesnost a to kvůli možnosti vynechání více oprávněných objektů. Na druhou stranu hodnoty, které dávají velmi dobré přibližnosti, odpovídají dražšímu provedení dotazu [35].

4.8.2 Míry výkonu

Hodnocení provedení algoritmů přibližného hledání podobností se zaměřuje na zlepšení efektivity a přesnosti přibližných výsledků a to kvůli přirozenému kompromisu mezi nimi. Značné zlepšení v efektivitě naproti preciznosti hledání podobnosti je typicky získáno na úkor přesnosti ve výsledcích. Pro porovnání různých algoritmů přibližného podobnostního vyhledávání je důležité znát vztah mezi těmito dvěmi mírami.

Zlepšení efektivity (IE) algoritmu přibližného vyhledávání s ohledem na precizní algoritmus je vyjádřeno jako cenový poměr provedení precizního a přibližného dotazu. Formálně je to definováno:

$$IE = \frac{\cos t(q)}{\cos t^A(q)}, \tag{4.28}$$

kde *cos t* a *cos* t^A značí počet přístupů na disk pro precizní a přibližné provedení dotazu *q*, v tomto pořadí, který bude buď R(q, r) nebo kNN(q). Např. zlepšení efektivity IE = 10 znamená, že přibližné provedení je desetkrát rychlejší než precizní. Vyhledávací náklady by mohly být alternativně měřeny počtem vzdálenostních výpočtů, ale experimenty demonstrují, že tyto dvě hodnoty jsou silně souvztažné.

Další míra ke stanovení kvality vyhledávání přibližného nejbližšího souseda je *relativní chyba nad vzdálenostmi*, navržená v [2]. Relativní chyba na vzdálenostech (*ED*) je definována jako

$$ED = \frac{d(o^A,q) - d(o^N,q)}{d(o^N,q)} = \frac{d(o^A,q)}{d(o^N,q)} - 1\ , \tag{4.29}$$

kde o^A je přibližný nejbližší soused a o^N je aktuální nejbližší soused. Relativní chyba nad vzdálenostmi měří kvalitu přibližnosti porovnáním vzdálenosti přibližného nejbližšího souseda s

aktuálním nejbližším sousedem z dotazovacího objektu. Toto může být snadno zobecněno do případu i-tého nejbližšího souseda tímto způsobem:

$$ED_i = \frac{d(o_i^A, q)}{d(o_i^N, q)} - 1 \ . \tag{4.30}$$

Předpokládejme, že vzdálenost mezi prvním a druhým aktuálním nejbližším sousedem je velká a taky, že přibližný algoritmus mine prvního nejbližšího souseda o^N, a první přibližný nejbližší soused o^A je ve skutečnosti druhý nejbližší soused. V tomto případě, relativní chyba na vzdálenostech je vysoká, dokonce i když je vynechán jen jediný objekt.

5 Aplikace

Cílem této práce bylo implementovat vzdálenostní funkce neboli *metriky*, které by mohly vést ať už k rychlejšímu vyhledávání a nebo k lepším (relevantnějším) výsledkům a následně navrhnout a implementovat grafické uživatelské rozhraní (aplikaci), na kterém by se otestovala funkčnost a provedly experimenty pro porovnání metrik.

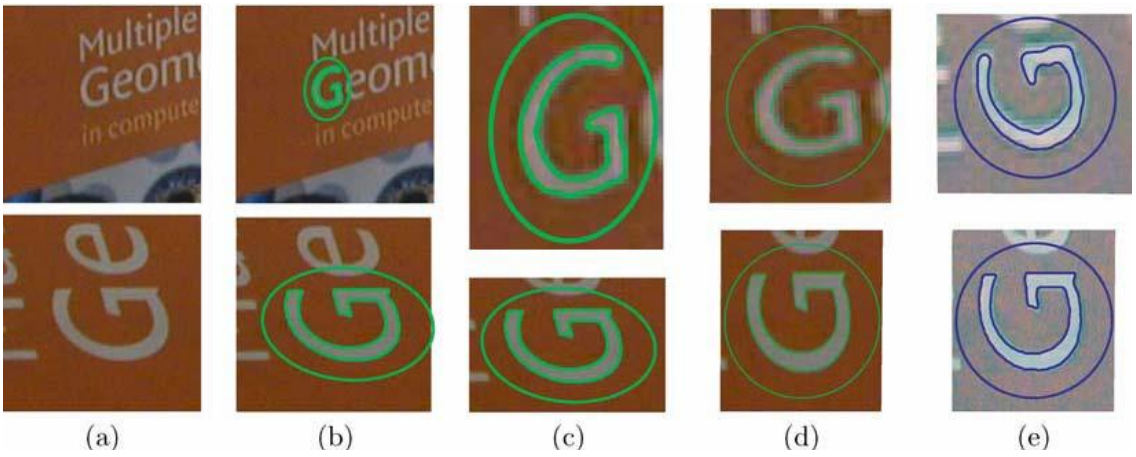
V této kapitole je popsána implementace všech kroků potřebných pro vyhledávání a provedené experimenty jsou v kapitole 6. Výsledná aplikace a celková funkčnost je dílem několika osob, které budou zmíněny u jednotlivých částí, na kterých se podílely.

5.1 Extrakce vizuálních rysů

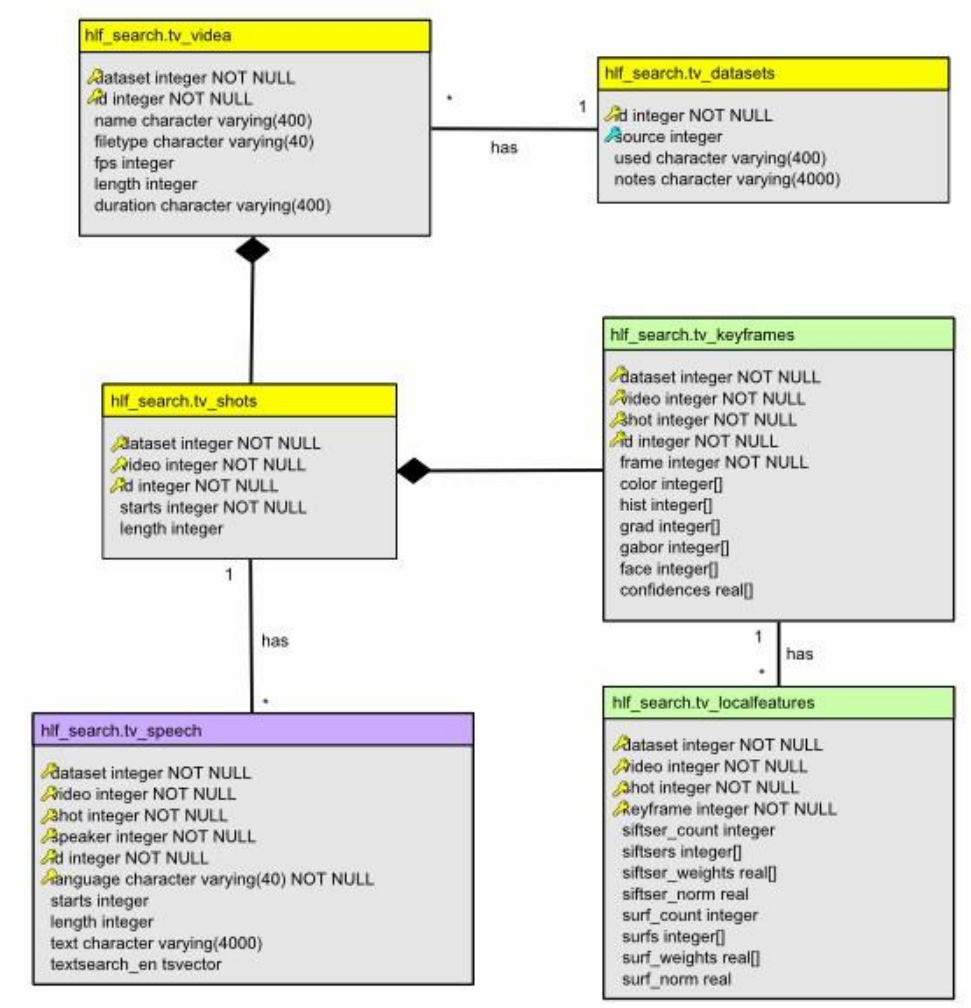
Prvním a to velmi důležitým krokem při vyhledávání je extrakce rysů ze zdrojových dat, v našem případě snímků videa. V námi vytvořené aplikaci je snímek videa (obraz) extrahován jako nějaký úsek videa mezi dvěma střihy tak, aby trval nejméně 1-2 s. Takový snímek je možné reprezentovat jedním (nebo několika) obrázky a to v závislosti na délce a množství změny ve snímku, popisem pohybu kamery a objektů, případně také popisem zvuku a automatickou extrakcí řeči, v případě dat použitých při evaluaci TRECVid. Výsledkem procesu extrakce je vektor rysů (vektor číselných hodnot), podle kterého se provádí vyhledávání (výpočtem vzdálenosti).

Implementaci extrakce rysů vyřešil vedoucí této práce Ing. Petr Chmelař a jde o sadu funkcí implementovaných v programovacím jazyce C. Byla provedena jak extrakce globálních tak i lokálních rysů. V případě *globálních rysů*, implementace téměř odpovídá standardu Multimedia Content Description Interface (MPEG-7 [21]), dalším detektorem je AdaBoost detektor obličejů [34], jehož vektorem rysů (o délce 4) je celkový počet obličejů (počet velkých – portrét, středních – hlasatel zpráv a malých – lidé v davu) v daném snímku videa, což by jiným typem popisu spadalo spíše do lokálních rysů.

U *lokálních rysů* jsou použity dvě metody pro detekci rysů a to *Speeded Up Robust Features* (SURF [3]) pro detekci zajímavých bodů, založený na výpočtu determinantu Hessovy matice (druhých parciálních derivací) z integrálního obrazu a druhá metoda je *Maximaly Stable Extremal Regions* (MSER, [23]) pro nalezení spojitých komponent vhodně prahovaného obrazu tak, aby byly maximálně stabilní. Extrémní zde pak znamená, že všechny pixely uvnitř regionu mají intenzitu nižší (tmavší) nebo vyšší, než pixely na okraji regionu, jak lze vidět na obr. 5.1.



Obr 5.1 (a) Originální obraz, (b) nalezené regiony, (c) detail nalezeného regionu, (d) geometrická normalizace na kruh, (e) fotometrická a geometrická normalizace [22].



Obr 5.2 Diagram tříd.

Druhým krokem extrakce lokálních obrazových rysů je reprezentace a to včetně jejich okolí. Na deskriptor jsou kladeny stejné požadavky jako na detektor – aby si odpovídaly stejné oblasti v různých obrázcích tak, jak je ilustrováno na obrázku 5.1. Překvapivě nejlépe zpracovaný popis obou kroků včetně přehledu jednotlivých metod jsme nalezli na Wikipedii [41], kompletní referencí jsou pak stránky [38], obsahující také referenční (použitý) software.

Pro popis lokálních rysů byl opět použit SURF, obsahující metodu založenou na Haarově vlnkové transformaci, která je obdobou metody nazvané *Scale Invariant Feature Transform* (SIFT, [22]), který využíváme pro popis regionů nalezených pomocí MSER. SIFT zachycuje určitou informaci o (elipsovitém okolí) bodu zájmu (středu regionu) pomocí histogramu lokálně orientovaných gradientů [17].

5.2 Vstupní data

Vstupem pro naši aplikaci byly vektory rysů extrahované ze snímků všech videí z vývojové datové sady TRECVid 2008 způsobem popsaným v sekci 5.1, které jsou uloženy v databázi „trecvid“ na školním serveru *minerva2.fit.vutbr.cz*. Výsledná aplikace však nevyužívá tuto databázi, ale databázi „trecvid“ na serveru *pcsocrates1.fit.vutbr.cz*. Jde o postrelační databázi PostgreSQL, kde vektory rysů jsou reprezentovány jako pole integerů.

Na obr. 5.2 je zobrazen diagram tříd modelovaný pomocí jazyka UML. Nejdůležitějšími tabulkami vytvořené aplikace jsou *hlf_search.tv_keyframe*s obsahující vektory globálních rysů a to *color* (barevnost dle definice TRECVid), *hist* (barevnostní histogram), *grad* (gradienty), *gabor* (textury) a *face* (obličeje na snímku), *hlf_search.tv_localfeatures* s vektory lokálních rysů extrahovaných jak pomocí metody MSER/Sift (atribut *siftsers*) tak i SURF (atribut *surfs*) a tabulka *hlf_search.tv_speech*, která slouží k textovému vyhledávání, tedy obsahuje textové popisy jednotlivých snímků (Automatic Speech Recognition).

Pro urychlení procesu vyhledávání, byla provedena indexace vektorů rysů a to zobecněným invertovaným indexem (Generalized Inverted Index – GIN).

5.3 Vzdálenostní funkce

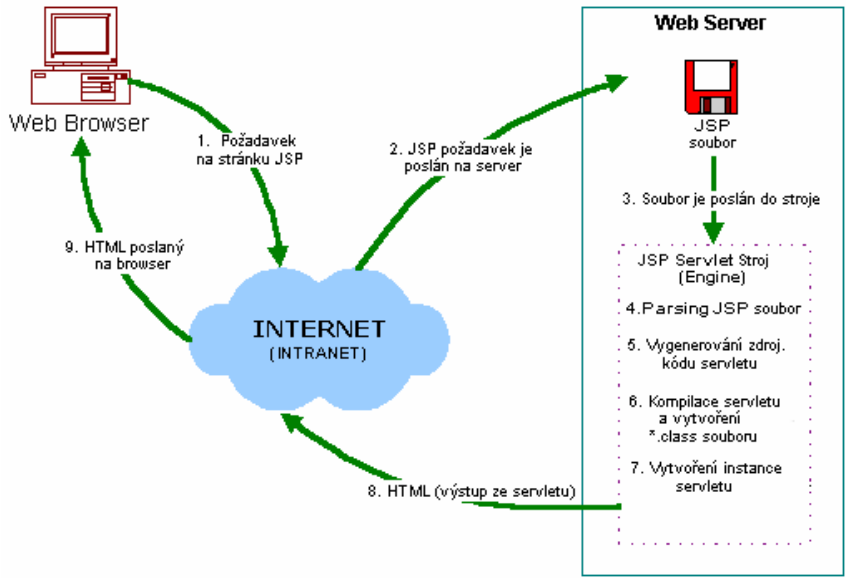
Jednotlivé vzdálenostní funkce byly implementovány v programovacím jazyce C, které následně byly vytvořeny jako funkce v databázi a volány pomocí SQL dotazů. Příklady kódu jazyka C a následné vytvoření funkcí v databázi PostgreSQL lze najít v dokumentaci [36].

Před vypracováním této práce Ing. Petr Chmelař implementoval Euklidovu vzdálenost pro vyhledávání podle globálních rysů, kosinovou větu a binární porovnání pro vyhledávání podle lokálních rysů, které byly použity při implementaci aplikace.

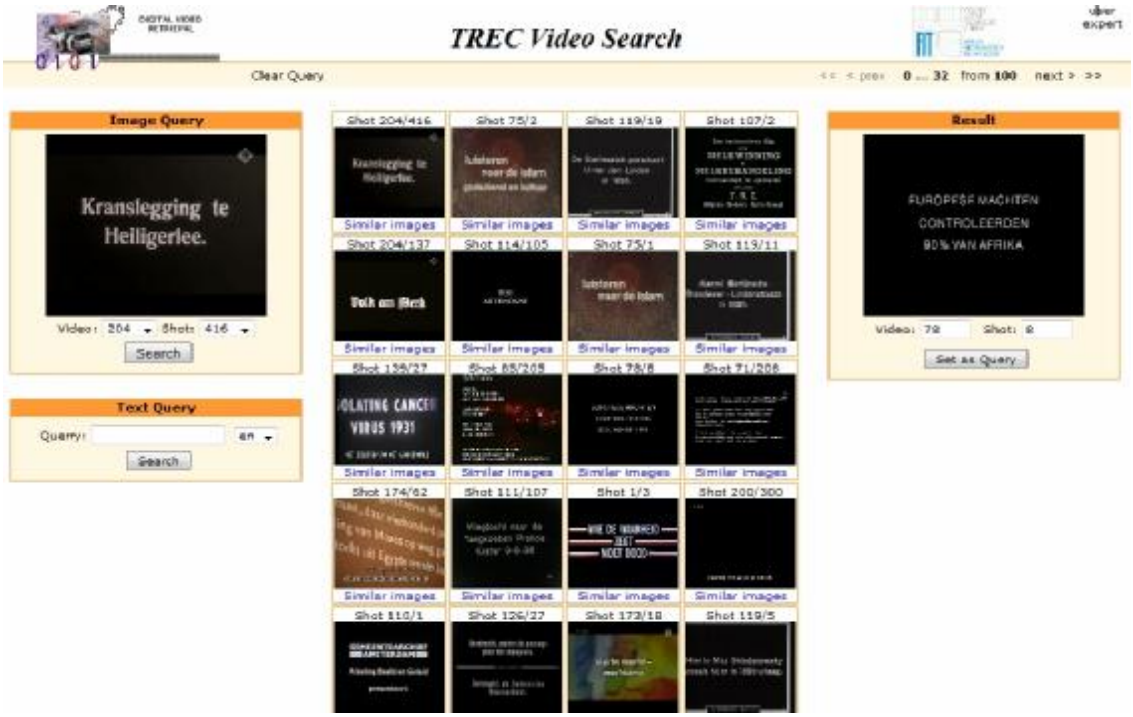
Jako další vzdálenostní funkce byla implementována Manhattanská vzdálenost, Chebyshevova vzdálenost a Mahalanobisova vzdálenost, popsány v sekci 4.2. Všechny tyto metriky byly, jak již jsem zmínil, implementovány v programovacím jazyce C, pouze pro výpočet Mahalanobisovy vzdálenosti je nutná směrodatná odchylka, která byla spočtena funkcí *standard_deviation(varchar)*, která byla vytvořena v programovacím jazyce PL/pgSQL. Dle vzorce 5.1, tato funkce vypočte směrodatnou odchylku toho sloupce tabulky *hlf_search.tv_keyframes*, který parametr je předán, tzn. parametr může být color, hist, grad, gabor a face. Výsledek této funkce je následně uložen do tabulky *hlf_search.tv_keyframes_aggr*.

$$s = \sqrt{\frac{\sum_{i=1}^{N} x_i^2}{N} - j^2}, \tag{5.1}$$

kde σ je směrodatná odchylka, N počet hodnot a φ je průměr všech hodnot.



Obr 5.3 Ukázka architektury JSP [5].



Obr 5.4 Ukázka aplikace v uživatelském režimu.

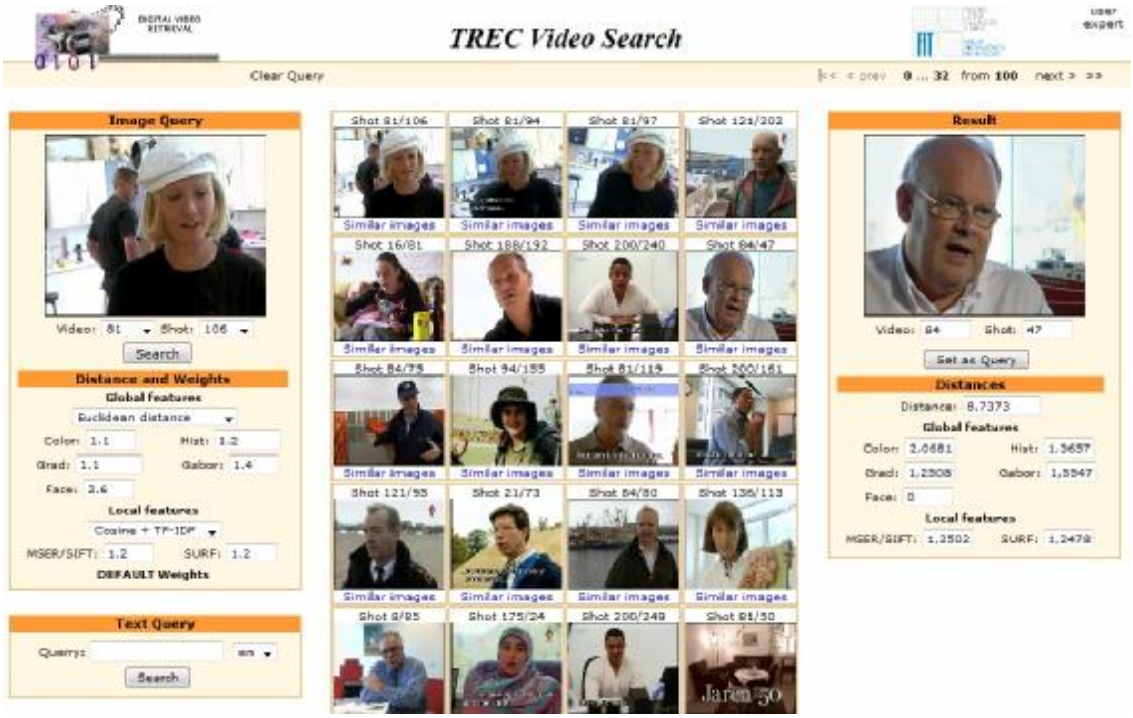
Další možností výpočtu vzdálenosti by mohlo být využití *Fischerova lineárního diskriminantu*, popsaného např.v [43]. Postup výpočtu byl nastudován a nejprve to měla být další metoda, podle které by se provádělo porovnání snímků, avšak nastaly komplikace při implementaci a to především při linkování některých velmi důležitých knihoven potřebných pro výpočet. Po konzultaci s vedoucím jsme se rozhodli danou metodu neimplementovat, protože nebylo jisté, zda-li by tato metoda byla v některém ohledu lepší než doposud implementované.

5.4 Grafické uživatelské rozhraní

Grafické uživatelské rozhraní je implementováno jako Java Server Pages (JSP). Je to zajímavý způsob, jak skloubit servlety, které jsou výkonově mohutné, ale programátorsky na vyšší úrovni, s jednoduchým jazykem, který je většině programátorů blízký a přitom dostatečně silný [31]. Architektura JSP je zobrazena na obr. 5.2.

Aplikace má dva režimy zobrazení. Prvním je uživatelský (ukázka na obr. 5.4), který je dosti intuitivní a jednoduchý i pro uživatele, který nebude nijak obeznámen s problematikou vyhledávání, tzn. že uživatel si může pouze vybrat vzorový obraz, který bude porovnáván s ostatními snímky a zobrazit výsledky dotazu nebo provést textové vyhledávání. Z uživatelského režimu lze velmi jednoduše přepnout do expertního režimu (samozřejmě to platí i obráceně), který je rozšířen o možnost nastavování vah jednotlivých atributů a výběr metriky použité při procesu vyhledávání. Dalším rozšířením je výpis vzdáleností u jednotlivých snímků zobrazených jako výsledek. Stejně jako v uživatelském režimu i v expertním je možné textové vyhledávání.

Před samotným vyhledáváním je nutné nejprve nastavit tzv. dotazový snímek (Image Query), tedy obrázek, který bude porovnáván s ostatními v databázi. Výběr lze provést z již zobrazených snímků, při spuštění aplikace se zobrazí sto náhodných obrázků nebo pomocí *comboboxů*.



Obr 5.5 Ukázka aplikace v expertním režimu.

V expertním režimu (ukázka na obr. 5.5) je pak potřeba nastavit vzdálenostní funkci a váhy atributů, které chceme aby byly brány v úvahu při procesu vyhledávání. Vyhledávání pak spustí SQL dotaz pouze s atributy, s nastavenými nenulovými váhami a dotaz vrátí sto snímků seřazených podle vzdálenosti od dotazového snímku a to vzestupně. Tyto snímky jsou pak v náhledech zobrazeny uživateli. U každého zobrazeného náhledu je pak možné jedním kliknutím provést nové vyhledání dle daného snímku se stejnými nastavenými váhami, jak byl snímek vyhledán.

K výsledné aplikaci přispěl i student Jakub Niec, který při své bakalářské práci navrhnul a implementoval aplikaci pro zobrazování náhledů snímků, která však nebyla plně funkční a dále implementoval připojení k databázi a textové vyhledávání, které bylo v této práci upraveno. Jakub Niec však svou bakalářskou práci nedokončil a proto byly výše zmiňované části aplikace použity v této práci.

5.5 SQL dotaz (vyhledávání)

Proces vyhledávání je prováděn jako jeden SQL dotaz, který využívá vzdálenostní funkce vytvořené v databázi.

```
SELECT dataset, video, shot,
        weight_feature * distance_function(query.feature, all.feature) AS distance_feature
        ( weight_feature * distance_function(query.feature, all.feature) ) AS dist
FROM  table1 AS all, table2 AS query
WHERE dataset=dataset
ORDER BY dist ASC
LIMIT 100";
```

Obr 5.6 Obecný SQL dotaz provádějící vyhledávání.

Tento dotaz tvoří funkce „*public String searchQuery(int dataset, int video, int shot, int metricG, int metricL, WeightsFormBean weightsForm)*“, kde parametry dataset, video a shot udávájí dotazový snímek, který je porovnáván s ostatními snímky v databázi, parametr metricG/metricL určuje jakou vzdálenostní funkci použít pro globální/lokální rysy a v parametru weightsForm jsou nastavené váhy všech atributů. Podle nastavených nenulových vah se počítají vzdálenosti a jejich součet. Na obr. 5.6 je zobrazen obecný dotaz, s nastavenou váhou pouze jednoho atributu *feature*. V příloze č. 3 je pak příklad dotazu s nastavenými všemi váhami.

6 Experimenty

V rámci evaluační soutěže TRECVid 2008 se naše fakulta v roce 2008 poprvé účastnila úloh detekce kopií (Content-based copy detection pilot) a vyhledávání (Search) ve videu. Pro tyto úlohy byly použity metody extrakce obrazových rysů (globální i lokální), indexace a vyhledávání principiálně založené na klasickém vyhledávání informací (Information Retrieval) s tím rozdílem, že vyhledávání je více zaměřené na lidském vnímání multimediálních dat a hledání je provedeno také v datech z úlohy HLF (High-Level Feature extraction) a v textu videa (ASR, Translation). TRECVid je od roku 2003 série konferencí sponzorovaná NIST (National Institute of Standards and Technology), kdy se oddělila od Text REtrieval Conference (TREC) [18].

Stěžejní úlohy evaluace TRECVid jsou extrakce rysů vysoké úrovně a vyhledávání. Jak již bylo zmíněno dříve, extrakce rysů byla vyřešena vedoucím této práce Ing. Petrem Chmelařem, tudíž tato práce využívá již hotové vyextrahované vektory rysů a zaměřuje se pouze na vyhledávání v těchto datech.

Pro představu, množství zpracovávaných dat v rámci úloh klasifikace a vyhledávání TRECVid 2008 nalezneme např. v [17]. Datové sady obsahují 438 videí (200 hodin) určených pro vývoj aplikace a její testování, avšak v naší aplikaci se provádí experimenty pouze na datové sadě 821, která obsahuje 219 videí (100 hodin). Pro doplnění představy, z těchto 219 videí je extrahováno ca. 36 tis. řádků globálních rysů (týkajících se celého snímku) a miliony lokálních rysů (týkající se jen určitých částí snímku), které jsou dále clusterovány [17] a TF-IDF normalizovány.

TRECVid 2008 obsahuje sadu (přesně čtyřiceti osmi) dotazů [39], ze kterých bylo vybráno několik z nich a na těch byly provedeny experimenty pro porovnání implementovaných metrik.

6.1 Postup experimentů

Prvním krokem experimentu vždy musí být nějaký dotaz jako např. „Nalezněte snímky s osobou otevírající dveře“ nebo „Nalezněte snímky s jedním nebo více lidmi, s jedním nebo více koňmi.“. U každého dotazu je v [39] i několik vzorových snímků, resp. časově ohraničených částí videa, které lze použít jako obraz, který se porovnává s ostatními v databázi. Tvar tohoto zápisu v [39] je ve formátu „*<videoExample src="nazev_videa" start="pocatecni_cas" stop="konecny_cas">*“. V aplikaci byla implementována funkce ***TimeToShot()***, která je zobrazena pouze v expertním režimu jako Video Query. Tato funkce vrací po zadání výše uvedeného formátu, číslo videa a středního snímku, který odpovídá danému popisu. Takový snímek následně nastavím jako dotazový.

Po nastavení dotazového snímku provedu vyhledávání pro všechny implementované metriky a po jednotlivých atributech, tzn. že všechny váhy jsou nastaveny na nulu, pouze atribut, podle kterého se vyhledává je nastaven na jedničku. Z použitých atributů vyplývá, že provedu dohromady patnáct vyhledávání pro globální rysy, neboť máme pět atributů (color, hist, grad, gabor a face) a tři vzdálenostní funkce (euklidova, chebyshevova a mahalanobisova) a další čtyři vyhledávání pro lokální rysy, neboť máme dva atributy (MSER/Sift a SURF) a dvě metody porovnání (kosinová a binární). U každého jednotlivého vyhledání je potřeba zapsat si čas, za jak dlouho byl dotaz proveden, kolik relevantních snímků bylo nalezeno v prvních 64 výsledcích (určení relevance je dle mého subjektivního názoru) a vzdálenost šedesátého čtvrtého a stého výsledku od dotazového snímku. Po zapsání všech důležitých hodnot provedu výpočet přesnosti a *průměrné přesnosti úplnosti a přesnosti* (popsáno v sekci 2.2) všech vyhledávání. Jako počet relevantních výsledků, které by měly být vráceny, jsem použil číslo 64. Vše jsem provedl dle projektu TRECVid pouze s tím rozdílem, že počet relevantních výsledků jsem určil sám, což by neměl být problém vzhledem k tomu, že toto číslo používám pouze pro porovnání přesnosti a výpočtu AP neboli průměrné přesnosti úplnosti a

přesnosti. Porovnání přesnosti a úplnosti nemá význam provádět, neboť jako počet vrácených výsledků, které by měly být vráceny jsem použil číslo 64, tzn. že hodnoty přesnosti a úplnosti budou shodné. Ze získaných a zapsaných výsledků vytvořím grafy pro viditelnější porovnání.

Posledním krokem je pak provedení vyhledávání s nastavenými váhami u všech atributů. Váhy těchto atributů musí být navrženy vzhledem k výsledným vzdálenostem všech atributů a především pak vzhledem k výsledné relevanci jednotlivých atributů. Nejprve sečteme relevantní výsledky všech atributů u jednotlivých metrik a z výsledných hodnot určíme, kterou metriku použijeme pro globální a kterou metodu pro lokální rysy závěrečného vyhledávání. Výpočet všech vah následně vypočítáme podle vzorce:

$$weight_f = \frac{d(q, o_{64})}{d(q, o_{100})} \times \frac{TP_f}{\sum_{i \in F} TP_i}, \tag{6.1}$$

kde $weight_f$ je výsledná váha atributu *f*, $d(q, o_{64})$ a $d(q, o_{100})$ je vzdálenost šedesátého čtvrtého, resp. stého výsledku od dotazového snímku, TP_f udává počet relevantních výsledků vyhledávání s atributem *f* a $\sum_{i \in F} TP_i$ je součet relevantních výsledků vyhledávání, s jednotlivými atributy. Množina *F* obsahuje atributy color, hist, grad, gabor, face, sift a surf. Je nutné doplnit, že tento způsob nelze použít při počítání váhy *face*, neboť nejlepší výsledky (to v některých případech může znamenat třeba i tisíce snímků) budou mít vzdálenost nula a tím by byla váha také rovna nule. Proto je váha face vypočtena podle stejného vzorce jen s tím rozdílem, že podíl vzdáleností je nahrazen číslem jedna. Vzhledem k tomu, že výsledné váhy jsou vždy v intervalu nula až jedna, rozhodl jsem se všechny váhy vynásobit číslem 10 a zaokrouhlit na jedno desetinné místo.

Veškeré experimenty byly prováděny na školním serveru *pcsocrates1.fit.vutb.cz*, jehož konfigurace je Intel(R) Pentium(R) 4 CPU 2.53GHz, 768MB paměti a pevný disk 36GB. Tato konfigurace je dosti nevyhovující, avšak pro experimenty této práce nám tento server postačil.

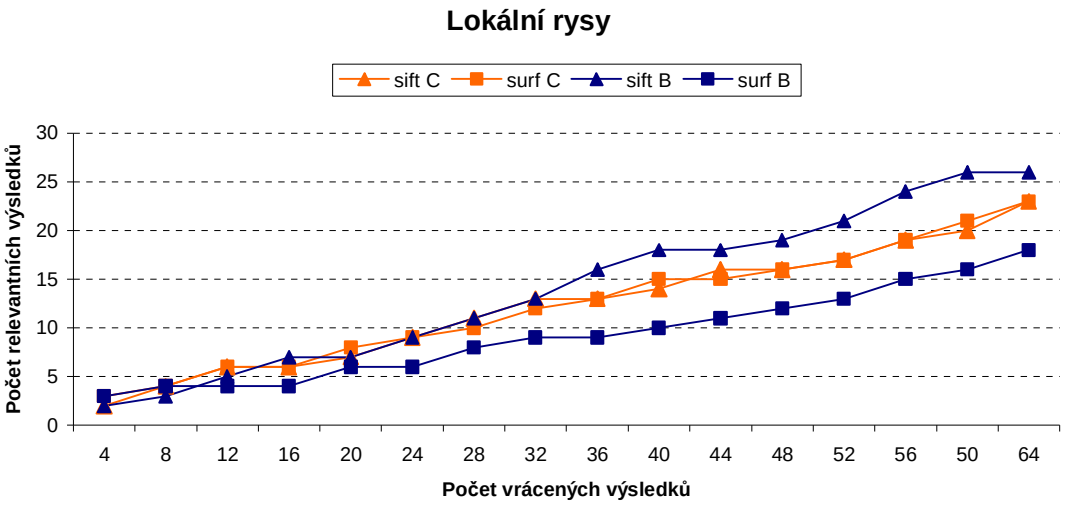
Na závěr bych se ještě zmínil o velikosti databázových tabulek a velikosti indexů jednotlivých tabulek. Velikosti tabulkek jsou, tv_keyframes - 277 MB, tv_localfeatures – 53 MB, tv_datasets – 8192 bytes, tv_videa – 72 kB, tv_shots – 7616 kB, tv_speech – 34 MB a velikosti indexů jsou, tv_keyframes 47 MB, tv_localfeatures – 182 MB, tv_datasets – 32 kB, tv_videa – 40 kB, tv_shots – 7352 kB a tv_speech – 9112 kB.

6.2 Výsledky

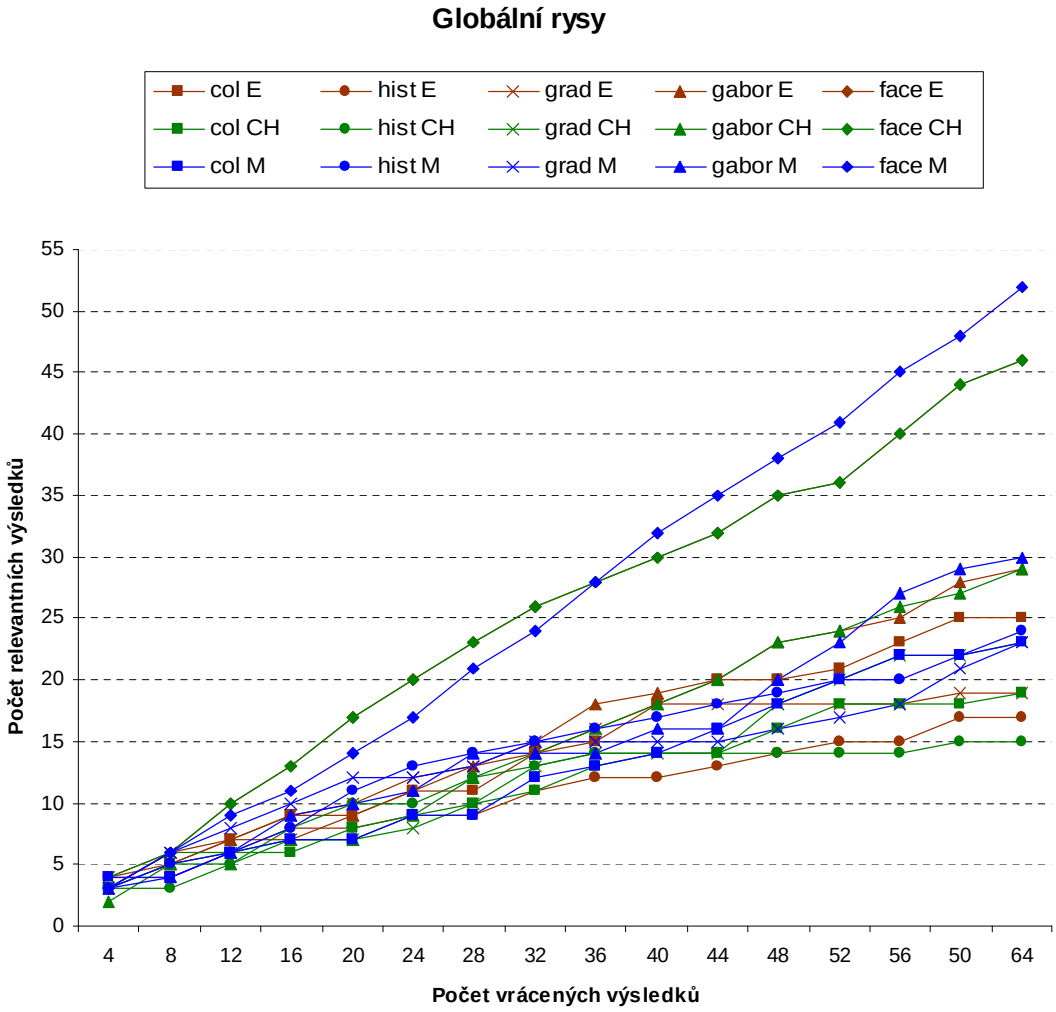
Jako první dotaz, jsem si vybral dotaz, který lze najít v [39] pod číslem „0267“:

```
<videoTopic num="0267">
   <textDescription text="Find shots with the camera zooming in on a person's face"/>
   <videoExample src="BG_36471.mpg" start="15m47.600s" stop="15m56.600s" desc="man's face"/>
   <videoExample src="BG_36471.mpg" start="07m10.280s" stop="07m20.000s" desc="woman's face"/>
   <videoExample src="BG_37322.mpg" start="12m22.760s" stop="12m29.600s" desc="outdoor shot"/>
   <videoExample src="BG_37940.mpg" start="03m30.720s" stop="03m33.560s" desc="child face"/>
   <videoExample src="BG_37940.mpg" start="14m56.560s" stop="14m59.000s" desc=""/>
</videoTopic>
```

Jde o dotaz „Nalezněte snímky s kamerou, která zaostřuje na lidskou tvář“ a jako dotazový snímek jsem použil druhý uvedený, zobrazující ženský obličej. Pomocí funkce TimeToShot() jsem zjistil, že jde o snímek číslo 24 z videa 175. Provedl jsem všechna vyhledávání dle sekce 6.1 a výsledky zobrazil do grafů. Porovnání lokálních rysů je zobrazeno na obr. 6.1 a porovnání globálních rysů na obr. 6.2. Tabulka se všemi hodnotami je k nahlédnutí v příloze č. 2.



Obr 6.1 Graf porovnání lokálních rysů u dotazu č. 267.



Obr 6.2 Graf porovnání globálních rysů u dotazu č. 267.

I když je podle mého názoru legenda grafů dosti intuitivní, raději ji více přiblížím. Každý popisek se skládá ze zkratky atributu a zkratky metriky (E – Euklidova, CH – Chebyshevova, M – Mahalanobisova, C – kosinová, B - binární), která byla použita při vyhledávání.

Z grafu globálních rysů lze vyčíst, že nejlepší výsledky měl atribut *face* a to s výpočty pomocí Mahalanobisovy metriky, Euklidova stejně jako Chebysheova metrika (jsou naprosto shodné) však nebyla nějak výrazně horší. Tento výsledek se určitě dal očekávat, protože se vyhledával obličej osoby a na této vlastnosti je atribut „face“ založen. Všechny ostatní atributy jsou výrazně horší. Mezi ty nejlepší pak patří atribut *gabor*, s využitím kterékoliv metriky, výsledky jsou téměř shodné. Nejhorším atributem se ukázal *hist* s výpočty Euklidovou nebo Chebyshevovou metrikou. U lokálních rysů byl nejlepším atributem *sift* s binárním porovnáním a nejhorším atributem *surf* opět s binárním porovnáním.

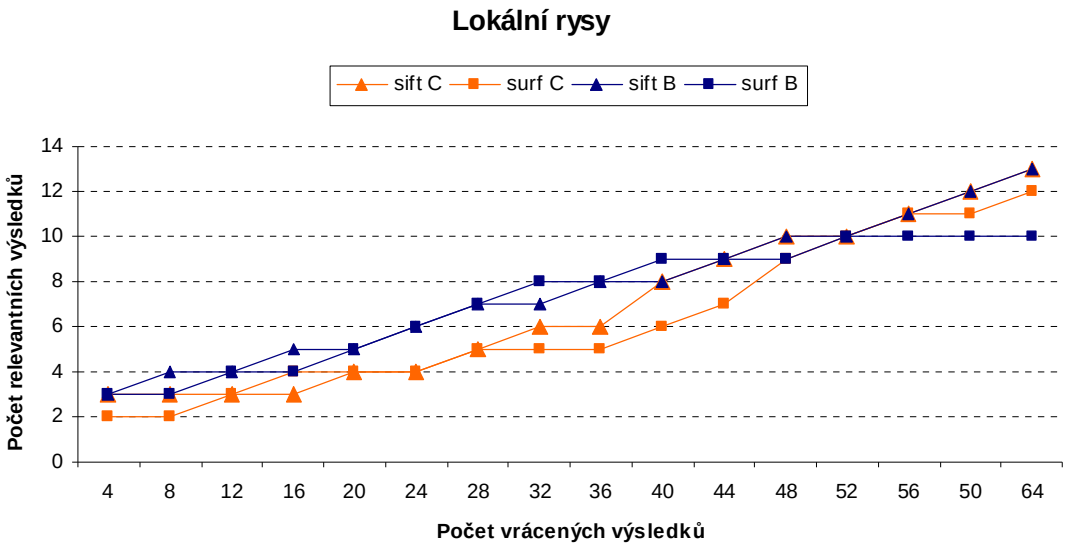
Z výsledků vyhledávání jsem vzhledem k získaným vzdálenostem a počtu relevantních výsledků došel k závěru, že pro tento dotaz by bylo nejvhodnější použít Mahalanobisovu metriku pro globální rysy a kosinovou větu pro lokální rysy. Podle postupu popsaného v sekci 6.1 jsem podle vzorce 6.1 vypočítal všechny váhy. Výsledné váhy jsou: color - 1.1, hist - 1.2, grad - 1.1, gabor - 1.4, face - 2.6, sift – 1.2 a surf – 1.2. S těmito váhami jsem provedl následné vyhledávání.

Výsledek tohoto vyhledávání byl nejlepší ze všech provedených. Z prvních 64 výsledků bylo 58 relevantních, což znamená s průměrnou přesností (viz. sekce 2.2) - *AP* : 85,6 procent. Pro představu, vyhledávání pouze podle jednotlivých atributů globálních rysů byl dotaz proveden přibližně za jednu vteřinu, u lokálních rysů, atributu surf, byl dotaz dokončen za necelé 3 vteřiny. Při nastavení všech atributů, dotaz vrátil výsledky za přibližně 17 vteřin. Tato doba je ovlivněna částečně výkonem počítače, ale především velkým množstvím výpočtů, které se musí provést.

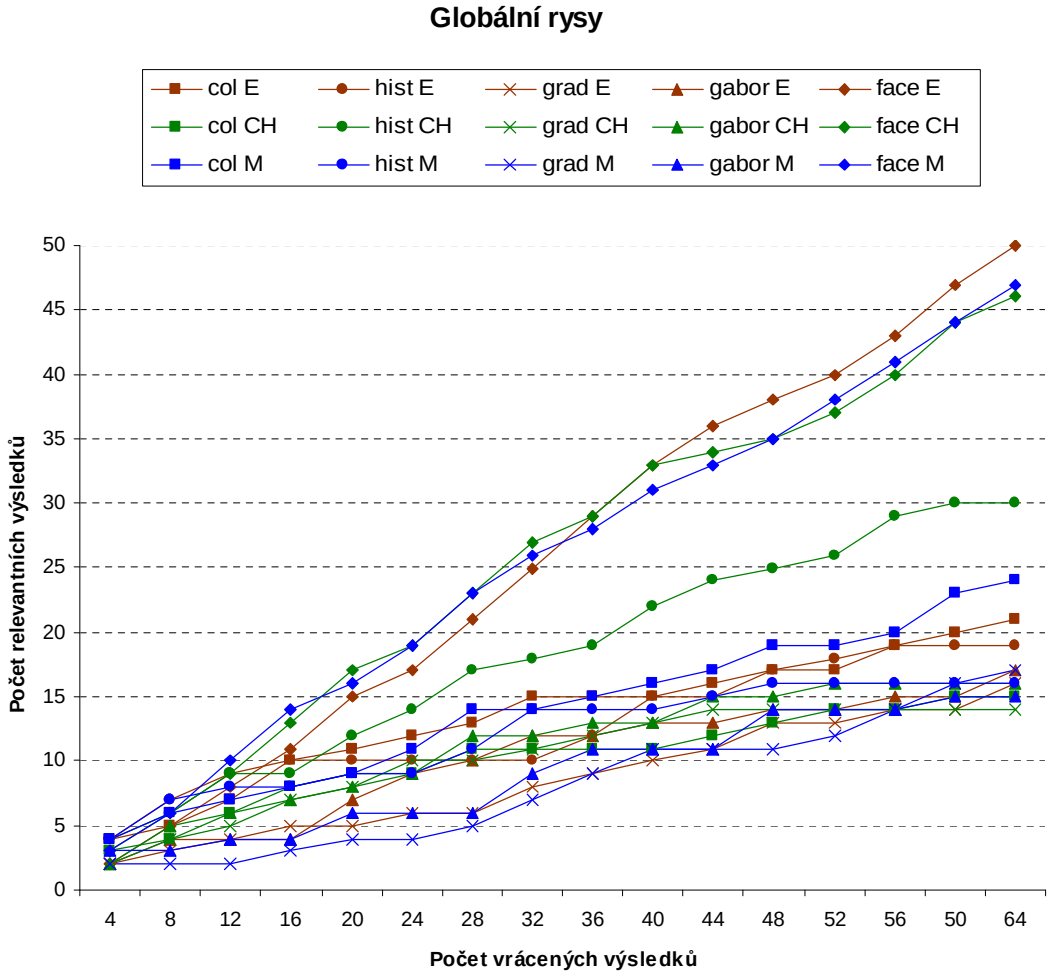
Jako druhý dotaz, jsem si vybral dotaz, který lze najít v [39] pod číslem „*0265*“:

```
<videoTopic num="0265">
  <textDescription text="Find shots of a man talking to the camera in an interview located indoors - no other people visible"/>
  <videoExample src="BG_37967.mpg" start="03m39.920s" stop="03m42.920s" desc=""/>
  <videoExample src="BG_37967.mpg" start="35m40.280s" stop="35m43.880s" desc=""/>
  <videoExample src="BG_35108.mpg" start="11m11.040s" stop="11m15.800s" desc="man in a lab coat"/>
  <videoExample src="BG_38905.mpg" start="06m21.040s" stop="06m26.000s" desc="man sitting"/>
  <videoExample src="BG_38905.mpg" start="09m48.360s" stop="09m53.000s" desc=""/>
</videoTopic>
```

Jde o dotaz „Nalezněte snímky muže, mluvícího do kamery v interview točeném uvnitř – nesmí být vidět žádná další osoba“ a jako dotazový snímek jsem použil třetí uvedený, zobrazující muže v laboratorním plášti. Pomocí funkce TimeToShot() jsem zjistil, že jde o snímek číslo 67 z videa 146. Provedl jsem všechna vyhledávání dle sekce 6.1 a výsledky zobrazil do grafů. Porovnání lokálních rysů je zobrazeno na obr. 6.3 a porovnání globálních rysů na obr. 6.4. Tabulka se všemi hodnotami je k nahlédnutí v příloze č. 2.



Obr 6.3 Graf porovnání lokálních rysů u dotazu č. 265.



Obr 6.4 Graf porovnání globálních rysů u dotazu č. 265.

Z grafu globálních rysů lze vyčíst, že nejlepší výsledky měl atribut *face* a to s výpočty pomocí Euklidovy metriky, avšak Mahalanobisova i Chebysheova metrika jsou jen nepatrně horší. Tento výsledek se určitě dal také očekávat, protože stejně jako u prvního dotazu se vyhledával obličej osoby a na této vlastnosti je atribut „face“ založen. Všechny ostatní atributy jsou opět výrazně horší. Druhým nejlepším výsledkem byl atribut *hist* s výpočtem pomocí Chebyshevovy metriky a dále následoval atribut *color* s Mahalanobisovou metrikou a Euklidovou metrikou. Nejhorším atributem se ukázal *grad* s výpočtem Chebyshevovou metrikou. U lokálních rysů byl nejlepším atributem *sift* a nezáleží na metodě porovnání, neboť obě mají shodné výsledky, nejhorším atributem pak byl *surf* s binárním porovnáním.

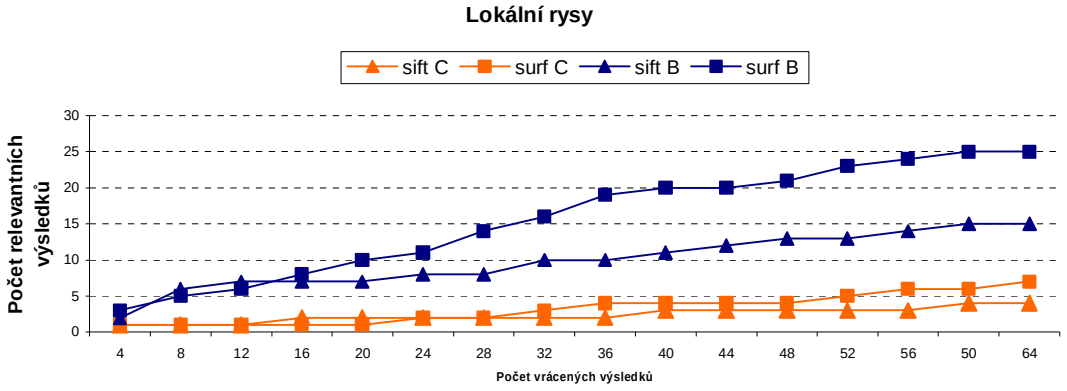
Z výsledků vyhledávání jsem vzhledem k získaným vzdálenostem a počtu relevantních výsledků došel k závěru, že pro tento dotaz by bylo nejvhodnější použít Euklidovu metriku pro globální rysy a kosinovou větu pro lokální rysy. Podle postupu popsaného v sekci 6.1 jsem podle vzorce 6.1 vypočítal všechny váhy. Výsledné váhy jsou: color - 1.4, hist - 1.2, grad - 1.0, gabor - 1.1, face - 3.4, sift – 0.9 a surf – 0.9. S těmito váhami jsem provedl následné vyhledávání.

Z prvních 64 výsledků bylo 48 relevantních, což znamená s průměrnou přesností - *AP* : 63,2 procent. Stejně jako u prvního dotazu je doba provádění vyhledávání přibližně 17 vteřin a je ovlivněna částečně výkonem počítače, ale především velkým množstvím výpočtů, které se musí provést.

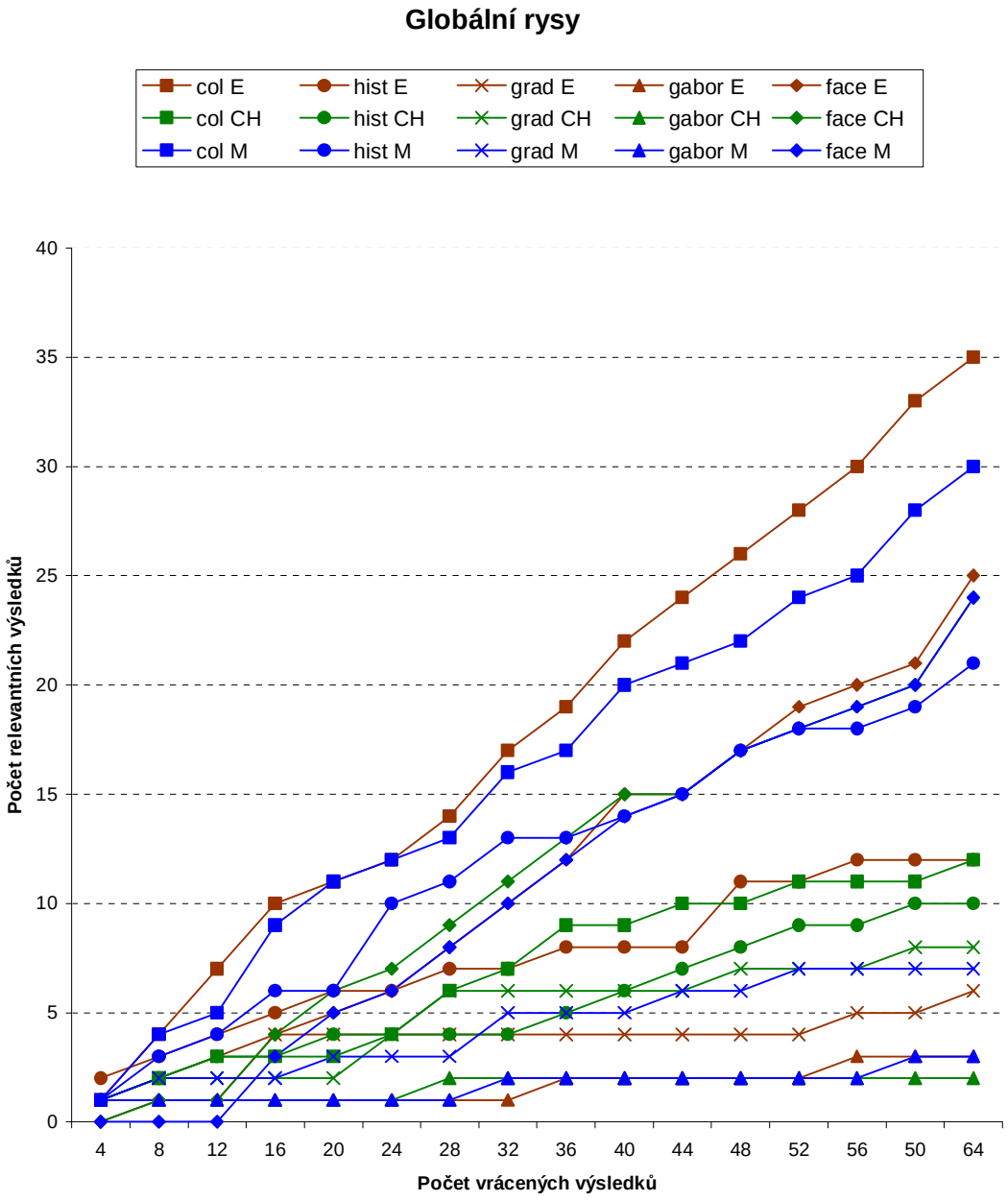
Jako třetí dotaz, jsem si vybral dotaz, který lze najít v [39] pod číslem „*0233*“:

```
<videoTopic num="0233">
  <textDescription text="Find shots of one or more black and white photographs, filling more than half of the frame area"/>
  <videoExample src="BG_35103.mpg" start="19m46.840s" stop="19m48.440s" desc="whole frame view"/>
  <videoExample src="BG_35059.mpg" start="01m29.320s" stop="01m32.320s" desc="whole frame view"/>
  <videoExample src="BG_35046.mpg" start="01m05.400s" stop="01m08.120s" desc="on desk view"/>
  <videoExample src="BG_37940.mpg" start="01m13.120s" stop="01m19.080s" desc="a set view"/>
  <videoExample src="BG_35108.mpg" start="03m19.440s" stop="03m23.960s" desc=""/>
</videoTopic>
```

Jde o dotaz „Nalezněte snímky jedné nebo více černo-bílých fotografií, vyplňující více než půlku oblasti snímku“ a jako dotazový snímek jsem použil první uvedený, zobrazující celou fotografii. Pomocí funkce TimeToShot() jsem zjistil, že jde o snímek číslo 96 z videa 145. Provedl jsem všechna vyhledávání dle sekce 6.1 a výsledky zobrazil do grafů. Porovnání lokálních rysů je zobrazeno na obr. 6.5 a porovnání globálních rysů na obr. 6.6. Tabulka se všemi hodnotami je k nahlédnutí v příloze č.2.



Obr 6.5 Graf porovnání lokálních rysů u dotazu č. 233.



Obr 6.6 Graf porovnání globálních rysů u dotazu č. 233.

Z grafu globálních rysů lze vyčíst, že nejlepší výsledky měl atribut *color* a to s výpočty pomocí Euklidovy metriky a Mahalanobisovy metriky. Tento výsledek se také dal očekávat, neboť hledáme černo-bílé fotografie, tzn. nejdůležitější jsou barvy na snímku, které jsou extrahovány ve vektoru color. Druhým nejlepším atributem byl *face* opět s výpočty pomocí Euklidovy a Mahalanobisovy metriky. Nejhorším atributem se ukázal *gabor*, což se také dalo očekávat, neboť textury byly v tomto hledání nepodstatné a druhým nejhorším pak atribut *grad* s použitím jakékoliv metriky. U lokálních rysů byl nejlepším atributem *surf* s binárním porovnáním a nejhorším atributem pak byl *sift* s výpočtem kosinovou větou.

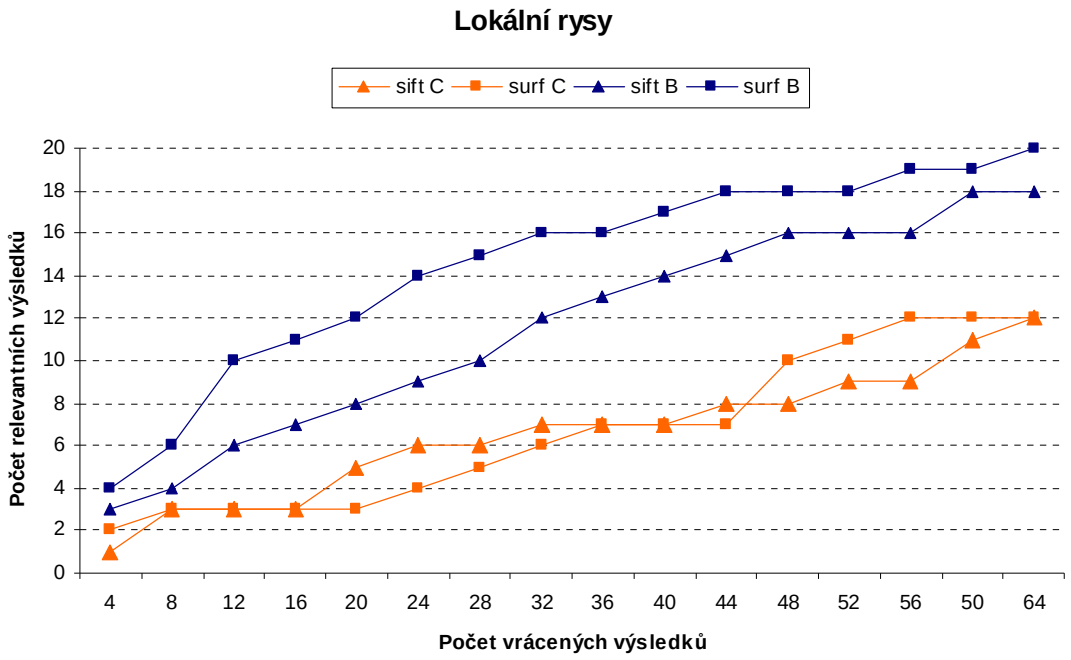
Z výsledků vyhledávání jsem vzhledem k získaným vzdálenostem a počtu relevantních výsledků došel k závěru, že pro tento dotaz by bylo nejvhodnější použít Mahalanobisovu metriku pro globální rysy a binární porovnání pro lokální rysy. Podle postupu popsaného v sekci 6.1 jsem podle vzorce 6.1 vypočítal všechny váhy. Výsledné váhy jsou: color - 2.5, hist - 1.7, grad - 0.6, gabor – 0.2, face – 2.1, sift – 0.6 a surf – 2.1. S těmito váhami jsem provedl následné vyhledávání.

Z prvních 64 výsledků bylo 23 relevantních, což znamená s průměrnou přesností – *AP*, pouze: 16,3 procent. Výsledek ukázal, že v tomto případě má relevantnější výsledky vyhledávání s použitím pouze atributu color než vyhledávání s nastavenými váhami všech atributů. Stejně jako u prvního dotazu je doba provádění vyhledávání ovlivněna částečně výkonem počítače, ale především velkým množstvím výpočtů, které se musí provést. Výsledná doba provádění dotazu byla necelých 10 vteřin.

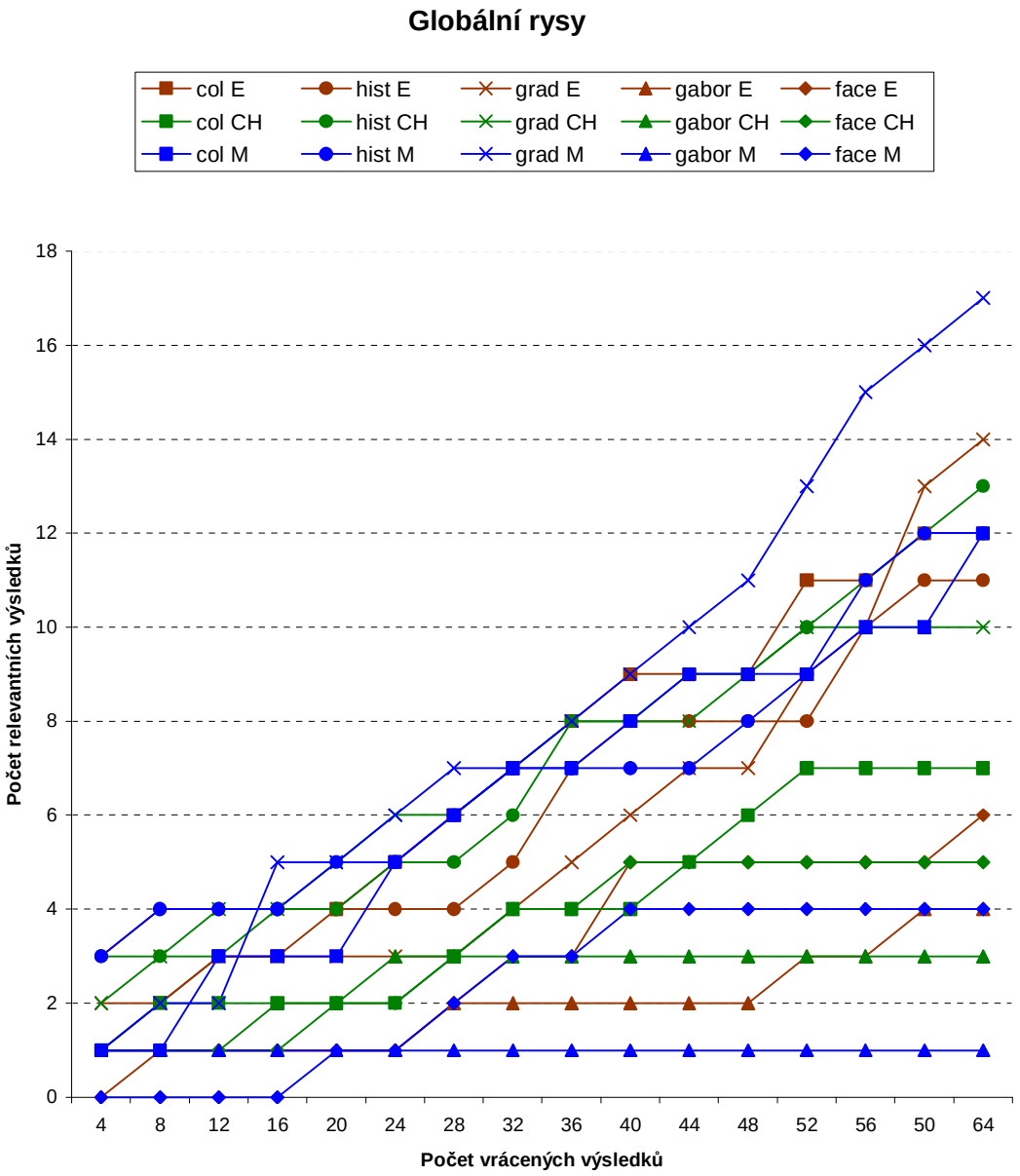
Jako čtvrtý dotaz, jsem si vybral dotaz, který lze najít v [39] pod číslem „*0248*“:

```
<videoTopic num="0248">
  <textDescription text="Find shots of a crowd of people, outdoors, filling more than half of the frame area"/>
   <videoExample src="BG_36643.mpg" start="27m06.480s" stop="27m12.800s" desc="front view"/>
   <videoExample src="BG_36729.mpg" start="16m19.920s" stop="16m23.240s" desc="view from above"/>
   <videoExample src="BG_38431.mpg" start="06m17.840s" stop="06m21.680s" desc="head height view"/>
   <videoExample src="BG_10241.mpg" start="11m25.760s" stop="11m32.000s" desc=""/>
   <videoExample src="BG_3097.mpg" start="11m09.040s" stop="11m14.000s" desc=""/>
</videoTopic>
```

Jde o dotaz „Nalezněte snímky davu lidí pohybujících se venku, zaplňující více než polovinu oblasti snímku“ a jako dotazový snímek jsem nepoužil žádný výše zmíněný, ale mnou vybraný snímek a jde o snímek číslo 48 z videa 149. Provedl jsem všechna vyhledávání dle sekce 6.1 a výsledky zobrazil do grafů. Porovnání lokálních rysů je zobrazeno na obr. 6.7 a porovnání globálních rysů na obr. 6.8. Tabulka se všemi hodnotami je k nahlédnutí v příloze č.2.



Obr 6.7 Graf porovnání lokálních rysů u dotazu č. 248.



Obr 6.8 Graf porovnání globálních rysů u dotazu č. 248.

Z grafu globálních rysů lze vyčíst, že nejlepší výsledky měl atribut ***grad*** a to s výpočty pomocí Mahalanobisovy metriky a Euklidovy metriky. Nejhorším atributem se ukázal ***gabor*** a druhým nejhorším pak atribut ***face*** s použitím jakékoliv metriky. U tohoto dotazu se ukázalo, že lokální rysy měly lepší výsledky než globální rysy. Nejlepším byl atribut ***surf*** s binárním porovnáním a nejhorší výsledky mělo porovnávání pomocí kosinové věty, nezávisle na atributu.

Z výsledků vyhledávání jsem vzhledem k získaným vzdálenostem a počtu relevantních výsledků došel k závěru, že pro tento dotaz by bylo nejvhodnější použít Euklidovu metriku pro globální rysy a binární porovnání pro lokální rysy. Podle postupu popsaného v sekci 6.1 jsem podle

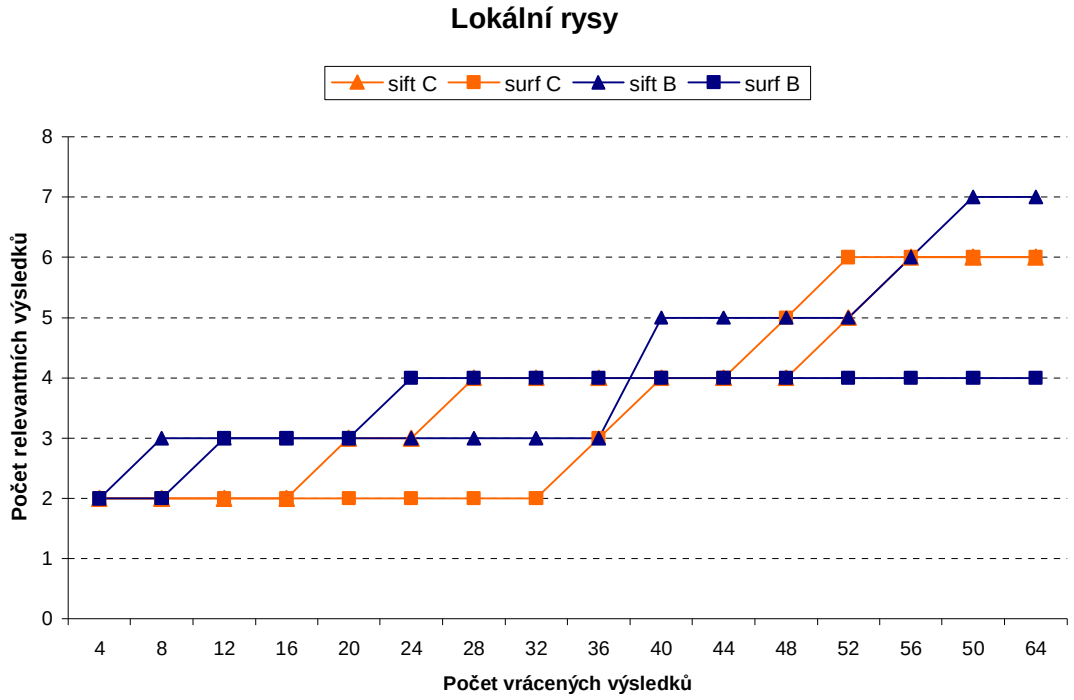
vzorce 6.1 vypočítal všechny váhy. Výsledné váhy jsou: color - 1.5, hist - 1.3, grad - 1.7, gabor – 0.5, face – 0.8, sift – 1.5 a surf – 2.5. S těmito váhami jsem provedl následné vyhledávání.

Z prvních 64 výsledků bylo 23 relevantních, což znamená s průměrnou přesností – *AP*, pouze: 17,2 procent. Výsledky ukázaly, že lokální rys, surf měl lepší relevanci než kterýkoliv globální rys, což nastalo pouze u tohoto dotazu. Stejně jako u prvního dotazu je doba provádění vyhledávání ovlivněna částečně výkonem počítače, ale především velkým množstvím výpočtů, které se musí provést. Výsledná doba provádění dotazu byla necelých 10 vteřin.

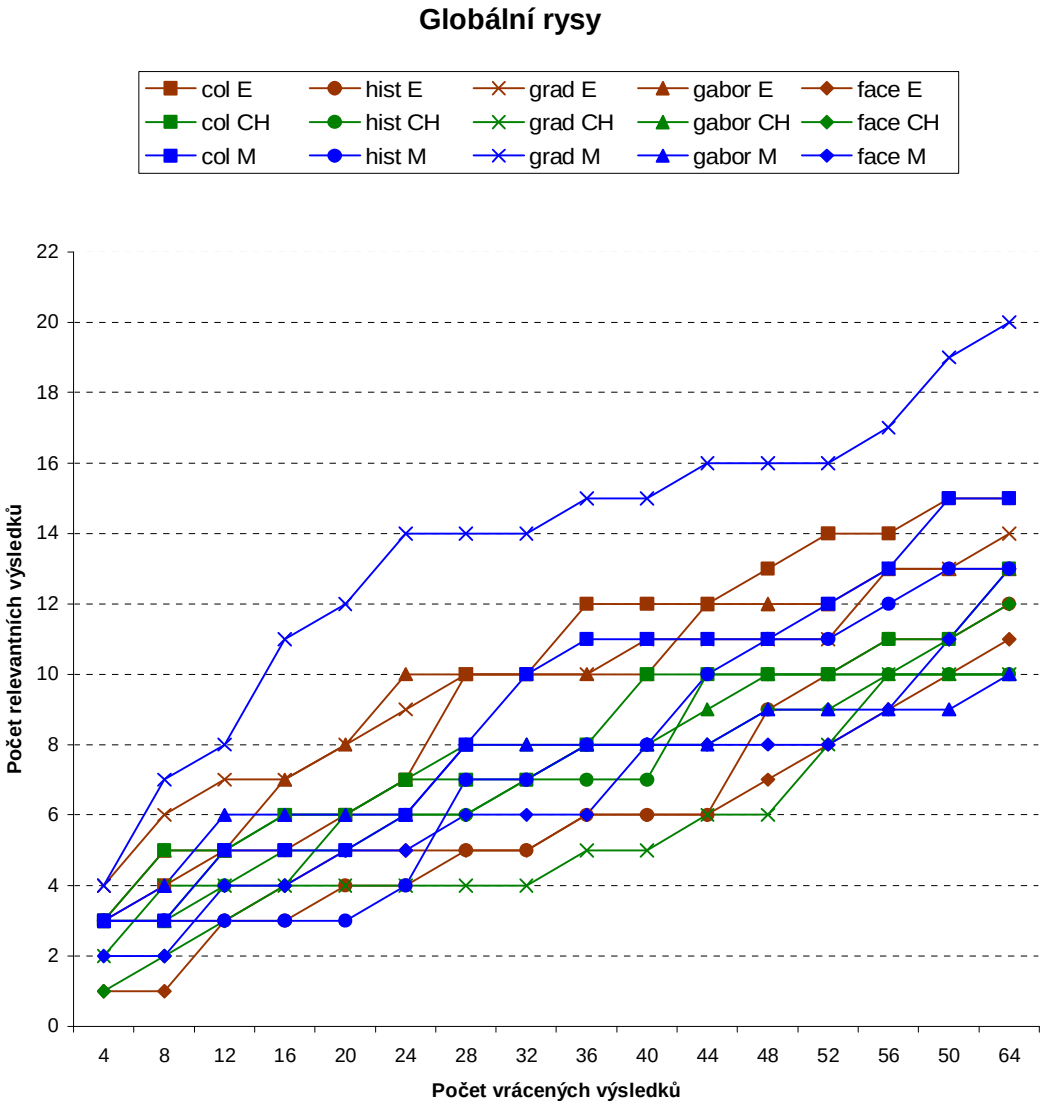
Jako poslední pátý dotaz, jsem si vybral dotaz, který lze najít v [39] pod číslem „*0237*“:

```
<videoTopic num="0237">
  <textDescription text="Find shots of a woman talking to the camera in an interview located indoors - no other people visible"/>
  <videoExample src="BG_37967.mpg" start="31m26.960s" stop="31m27.720s" desc="woman in office"/>
  <videoExample src="BG_37967.mpg" start="31m53.400s" stop="31m57.440s" desc="face only "/>
  <videoExample src="BG_35059.mpg" start="08m14.200s" stop="08m17.600s" desc="face only"/>
  <videoExample src="BG_38191.mpg" start="23m04.280s" stop="23m08.960s" desc="upper body visible"/>
  <videoExample src="BG_3097.mpg" start="06m02.720s" stop="06m06.200s" desc=""/>
</videoTopic>
```

Jde o dotaz „Nalezněte snímky ženy, mluvící do kamery v interview točeném uvnitř – nesmí být vidět žádná další osoba“ a jako dotazový snímek jsem použil první uvedený, zobrazující ženu v kanceláři. Pomocí funkce TimeToShot() jsem zjistil, že jde o snímek číslo 172 z videa 201. Provedl jsem všechna vyhledávání dle sekce 6.1 a výsledky zobrazil do grafů. Porovnání lokálních rysů je zobrazeno na obr. 6.9 a porovnání globálních rysů na obr. 6.10. Tabulka se všemi hodnotami je k nahlédnutí v příloze č. 2.



Obr 6.9 Graf porovnání lokálních rysů u dotazu č. 237.



Obr 6.10 Graf porovnání globálních rysů u dotazu č. 237.

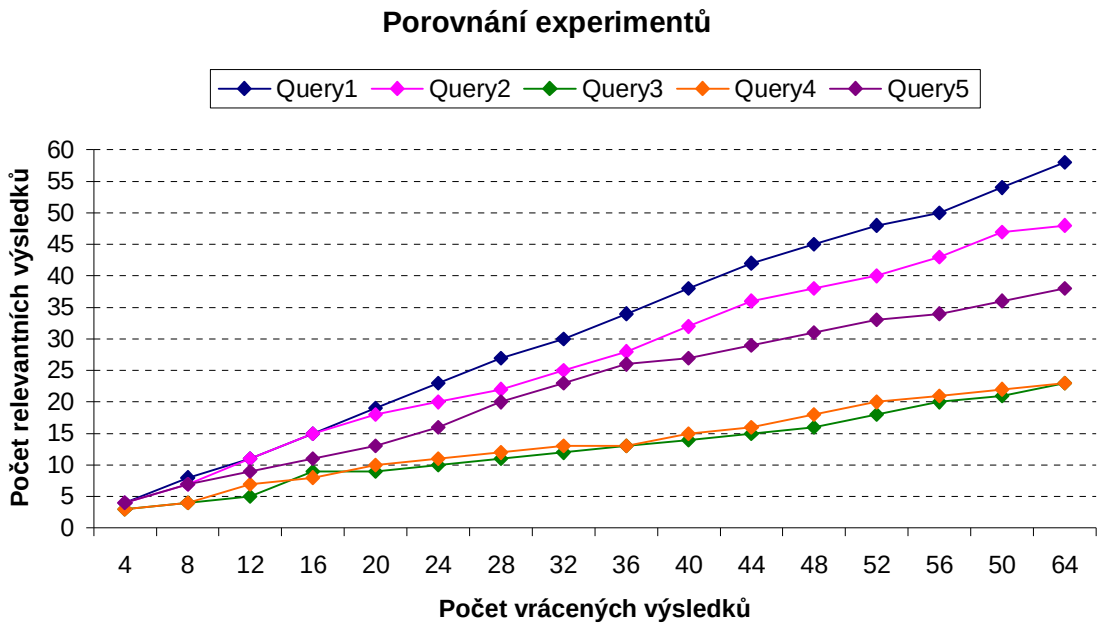
Z grafu lze vyčíst, že nejlepší výsledky měl atribut ***grad*** a to s výpočty pomocí Mahalanobisovy metriky. Tento výsledek je dosti překvapivý, protože se vyhledával obličej osoby a na této vlastnosti je založen atribut „face“. Druhým nejlepším výsledkem byl atribut ***color*** s výpočtem pomocí Mahalanobisovy metriky a Euklidovy metriky. Nejhoršími atributy se ukázaly ***grad, gabor a hist*** s výpočtem pomocí Chebyshevovy metriky spolu s atributem ***gabor*** s výpočtem pomocí Mahalanobisovy metriky. U lokálních rysů byl nejlepším atributem ***sift*** s binárním porovnáním a nejhorším atributem pak atribut ***surf*** opět s binárním porovnáním.

Z výsledků vyhledávání jsem vzhledem k získaným vzdálenostem a počtu relevantních výsledků došel k závěru, že pro tento dotaz by bylo nejvhodnější použít Mahalanobisovu metriku pro globální rysy a kosinovou větu pro lokální rysy. Podle postupu popsaného v sekci 6.1 jsem podle vzorce 6.1 vypočítal všechny váhy. Výsledné váhy jsou: color - 1.7, hist - 1.5, grad - 2.3, gabor – 1.1, face – 1.5, sift – 0.7 a surf – 0.8. S těmito váhami jsem provedl následné vyhledávání.

Z prvních 64 výsledků bylo 38 relevantních, což znamená s průměrnou přesností – *AP*: 43 procent. Stejně jako u prvního a ostatních dotazů je doba provádění vyhledávání ovlivněna částečně výkonem počítače, ale především velkým množstvím výpočtů, které se musí provést. Výsledná doba provádění dotazu byla necelých 15 vteřin.

6.3 Zhodnocení výsledků

Výsledky předešlých vyhledávání (s nastavenými všemi váhami) jsem porovnal a zobrazil do grafu (obr. 6.11).



Obr 6.11 Graf porovnání všech provedených experimentů.

Z grafu je patrné, že nejlepší výsledky měly experimenty jedna (snímky s kamerou zaostřující na lidskou tvář), dva (snímky muže, mluvícího do kamery v interview točeném uvnitř) a pět (snímky ženy, mluvící do kamery v interview točeném uvnitř, nesmí být vidět žádná další osoba). Tyto výsledky jsou tak dobré především proto, že se vyhledával nějaký obličej a na této vlastnosti je založen atributu face, který obsahuje počty obličejů na snímku. První experiment je výrazně lepší proto, že dotaz zněl dosti obecně, měl obsahovat jakýkoliv přiblížený obličej na rozdíl od druhého experimentu, kde v dotazu stálo „mužský obličej“ a ještě musel být uvnitř nějaké budovy a nesměli být vidět další osoby. Třetí experiment byl jediný, u kterého vyhledávání s nastavením všech atributů vyšlo hůře než při vyhledávání pouze s některým jedním atributem, v tomto případě atributem color. Tento výsledek není až tak překvapivý, neboť se vyhledával černo-bílý snímek a v takovém případě jde především o barevnost obrázku. Poslední čtvrtý experiment byl zase jediný dotaz, u kterého vyšly lepší výsledky u lokálních rysů než u globálních rysů.

Ze všech provedených experimentů jsem došel k závěru, že nejlepší vzdálenostní funkcí pro globální rysy je Mahalanobisova vzdálenost. Výsledky s Euklidovou vzdáleností jsou však dosti srovnatelné a dalo by se očekávat, že při větším počtu experimentů, by tomu mohlo být jinak, proto považuji za nejlepší metriky jak Mahalanobisovu, tak i Euklidovu. Rozdíl těchto metrik je pak především v časech provádění, neboť při velmi rozměrných vektorech je doba provádění Mahalnobisovi vzdálenosti podstatně delší než u Euklidovy. Chebyshevova vzdálenost se pro

globální rysy ukázala jako nejhorší z použitých. U mnou vybraných experimentů lokálních rysů se ukázalo, že kosinová věta je téměř shodná s binárním porovnáním. Předpokládám však, že při větším počtu experimentů by se prokázalo, že kosinová věta je lepší metodou než binární porovnání.

Porovnání jednotlivých atributů je dosti obtížné, neboť vždy záleží na položeném dotazu. Atribut *face* je nejlepším pro dotazy, ve kterých se dotazujeme na osoby, např. „Muž stojící u dvěří“ nebo „Dvě osoby sedící u stolu“, atribut *color* zase pro dotazy ohledně barevnosti obrazu jako např. „Černo-bílá fotka“. U lokálních rysů se z experimentů ukázalo, že atribut SURF je celkově lepší než atribut SIFT, avšak pouze nepatrně a také záleží na položeném dotazu.

Z provedených experimentů jde vidět, že výsledky jednotlivých dotazů nejsou vůbec špatné, avšak otázkou je, jaké váhy nastavit obecně pro všechny typy dotazů. Možností je např. provést více experimentů, vzít výsledné váhy a provést průměr vah u jednotlivých atributů, což by mohlo být rozšíření či pokračování této práce. Další experimenty budou provedeny v NIST při evaluaci TRECVid 2009.

7 Závěr

Množství digitálních dat jako videa, audio záznamů nebo obrázků se v reálném světě stále zvětšuje a proto systémy pro uchovávání takových dat a následné vyhledávání z nich jsou opravdu nezbytné. Cílem vývoje této oblasti je snaha o dokonalejší a neomylnější způsoby prezentace těchto dat a to především vhledem k lidskému vnímání, tzn. aby při procesu vyhledávání byl opravdu nalezen např. snímek, na kterém je zobrazena loď projíždějící pod mostem.

Cílem této práce bylo prostudovat problematiku vyhledávání v multimodálních databázích, což znamenalo hlouběji proniknout do oblasti podobnostního vyhledávání založeného na metrických prostorech. V dnešní době již existuje řada metod používaných pro podobnostní vyhledávání, avšak pouze některé jsou využitelné v reálných systémech.

Jak již bylo zmíněno, důležitou otázkou je, jak prezentovat multimediální data, v našem případě snímky videa? U globálních rysů, popisující celý obrázek se ukázalo, že nejvhodnější je použít popis textur ve snímku, dle specifikace MPEG-7 a barevnost snímku, dle specifikace TRECVid a u lokálních rysů, popisující jen určité části snímku je velmi podstatné, aby určitý předmět či obličej na snímku byl extrakcí nalezen i na snímku s kamerou zabírající záběr pod jiným úhlem, či při naklonění hlavy s daným obličejem. Celkové zhodnocení výsledků experimentů je popsáno v sekci 6.3.

Implementace aplikace byla provedena jako Java Server Pages a výpočty vzdáleností byly implementovány v programovacím jazyce C, neboť provádění matematických výpočtů v jazyce C je rychlejší. Časy provádění některých dotazů při experimentech byly často dosti vysoké, což bylo způsobeno parametry serveru (vyřazeného PC), na kterém se veškeré výpočty vzdáleností prováděly. Tato práce i výsledná aplikace se zaměřuje pouze na obrazové vyhledávání, tzn. nepracuje s audio záznamy, klasifikace se také něřeší.

Vývoj oblasti vyhledávání informací jde stále kupředu a v posledních letech dosti rychle, především díky projektu TRECVid. Ještě nedávno neexistoval systém, který by plně odpovídal náročným uživatelským požadavkům a vždy dokázal vyhledat přesně to, co uživatel požaduje, avšak v dubnu roku 2009 spustil server Google své nové stránky „http://similar-images.googlelabs.com/“, které vyhledávájí podle obsahu i textového popisu dohromady. Podle vývoje této oblasti lze očekávat, že v budoucnu budou aplikace z toho oboru usnadňovat mnoho lidských činností.

Směr dalšího výzkumu a vývoje této práce by mohla být např. implementace dalších vzdálenostních funkcí, které by mohly urychlit proces vyhledávání nebo by mohly vracet relevantnější výsledky (např. *Fischerův lineární diskriminant*). Vzhledem k tomu, že množství použitelných vzdálenostních funkcí není příliš velké a není jistota, zda by výsledky byly výrazně lepší než doposud implementované zaměřil bych se spíše na využití technik na snížení množství počítaných vzdáleností, které jsou v této práci také popsány a to v sekci 4.6. Jako dalším rozšířením této práce by mohl být návrh některé indexové struktury, jiné než GIN, jako např. M-strom, který je pro metrické prostory často využíván pro urychlení přístupu k datům.

Literatura

[1] APOSTOLICO, A. and GALIL, Z. *Pattern Matching Algorithms.* 1997. Oxford University Press.

[2] ARYA, S., MOUNT, D. M., NETANYAHU, N. S., SILVERMAN, R., and WU, A. Y. *An optimal algorithm for approximate nearest neighbor searching in fixed dimensions*. 1998. ACM Press.

[3] BERKHIN, P. *A Survey of Clustering Data Mining Techniques*. 2006. p. 25-71, ISBN 978-3-540-28348-5.

[4] BOHM, C., BERCHTOLD, S., and KEIM, D. A. *Searching in high- dimensional spaces: Index structures for improving the performance ofmultimediadatabases.* 2001. ACM Computing Surveys (CSUR 2001), 33(3):322-373. ACM Press.

[5] BRANICKÝ, Marek. JavaServer Pages pro všechny. *Interval.cz* [online]. 2002 [cit. 2009-05-11]. Dostupný z WWW: <http://interval.cz/clanky/javaserver-pages-pro-vsechny/>.

[6] CIACCIA, P., MONTESI, D., PENZO, W., and TROMBETTA, A. *Imprecision and user preferences in multimedia queries: A generic algebraic approach.* 2000. Springer.

[7] CIACCIA, P., PATELLA, M., and ZEZULA, P. *Processing complex similarity queries with distance-based access methods*. 1998. Springer.

[8] CIACCIA, P. and PATELLA, M. *The M-tree: Processing complex multi-feature queries with just one index*. 2000.

[9] COBENA, G., ABITEBOUL, S., and MARIAN, A. *Detecting changes in XML documents*. 2002.

[10] FERHATOSMANOGLU, H., TUNCEL, E., AGRAWAL, D., and ABBADI, A. E. *Approximate nearest neighbor searching in multimedia databases*. 2001.

[11] GIRALDA, A. R. *Image Databases Indexing.* 2006. Vedoucí MT Zendulka, J.

[12] HEAPS, Chris, MILLS, Ci-Ci. *Inverted Index* [online]. 1996 [cit. 2009-05-08]. Dostupný z WWW: <http://www.python.org/workshops/1996-11/papers/InvertedIndex.html>.

[13] HJALTASON, G. R. and SAMET, H. *Incremental similarity search in multimedia databases*. 2000.

[14] HJALTASON, G. R. and SAMET, H. *Index-driven similarity search in metric spaces*. 2003. ACM Press.

[15] HUTTENLOCHER, D. P., KLANDERMAN, G. A., and RUCKLIDGE, W. J. *Comparing images using the Hausdorff distance.* 1993.

[16] CHMELAŘ, Petr. *Multimediální databáze* [online]. 2006 [cit. 2009-05-05]. Dostupný z WWW: <http://www.fit.vutbr.cz/study/courses/VPD/public/0506VPD-Chmelar.pdf>.

[17] CHMELAŘ, Petr, RUDOLFOVÁ, Ivana. *Shlukování založené na Voronoiově dláždění pro klasifikaci a vyhledávání ve videu*. 2008.

[18] CHMELAŘ, Petr. *Vyhledávání TRECVid 2008* [online]. 2008 [cit. 2009-05-11]. Dostupný z WWW: <http://www.fit.vutbr.cz/research/pubs/TR/2008/sem_uifs/s081020slidy1.pdf>.

[19] KRÁTKÝ, Michal, SKOPAL, Tomáš, SNÁŠEL, Václav. *Efektivní vyhledávání v kolekcích obrázků tváří* [online]. 2003 [cit. 2009-05-07]. Dostupný z WWW: <http://www.cs.vsb.cz/kratky/courses/2003-04/dis/reference/effface.pdf>.

[20] LEVENSHTEIN, V. I. *Binary codes capable of correcting spurious insertions and deletions of ones.* Problems of Information Transmission, 1:8-17. 1965. Kluwer Academic Publishers.

[21] MARTÍNEZ, José M., MALLORCA, Palma de. *ISO/IEC JTC1/SC29/WG11 MPEG-7 Overview (version 10)* [online]. 2004 [cit. 2009-05-06]. Dostupný z WWW: <http://www.chiariglione.org/mpeg/standards/mpeg-7/mpeg-7.htm>.

[22] MIKOLAJCZYK, K. *A Comparison of Affine Region Detectors*. International Journal of Computer Vision 65, no. 1 (November 13, 2005): 43-72.

[23] MIKOLAJCZYK, K. et al. *A Comparison of Affine Region Detectors*. 2005.

[24] OXFORD DIGITAL LIBRARY. *Metadata in the Oxford Digital Library* [online]. 2005 [cit. 2009-05-05]. Dostupný z WWW: <http://www.odl.ox.ac.uk/metadata.htm>.

[25] OXLEY, S. *Modelling the Capture Tudory for the Origin of Planetary Systéme*, New York, 1999.

[26] PÁNEK, Karel. Architektury a modely webových strojů. *LUPA : server o českém internetu* [online]. 2002 [cit. 2009-04-28]. Dostupný z WWW: <http://www.lupa.cz/clanky/architektury-a-modely-webovych-stroju/>.

[27] PATELLA, M. and CIACCIA, P. *Searching in metric spaces with user-defined and approximate distances*. 2002. ACM Press.

[28] PILECKÁ, Věra. VYHLEDÁVÁNÍ INFORMACÍ. *Ústav informačních studií a knihovnictví* [online]. 2006 [cit. 2009-04-28]. Dostupný z WWW: <uisk.ff.cuni.cz/dwn/1003/2061cs_CZ_REŠERŠNÍ%20STRATEGIE%2010.11.2006%20kombi.ppt>.

[29] SKOPAL, Tomáš, KOLOVRAT, Michal, SNÁŠEL, Václav. *Využití LSI a M-stromu pri indexování a vyhledávání obrázku* [online]. 2005 [cit. 2009-05-07]. 12 s. Dostupný z WWW: <http://siret.ms.mff.cuni.cz/skopal/pub/znal2005.pdf>.

[30] SKOPAL, Tomáš, KRÁTKÝ, Michal, SNÁŠEL, Václav. *Geometrické indexování a dotazování multimediálních dat* [online]. 2006 [cit. 2009-05-07]. Dostupný z WWW: <http://kebrt.webz.cz/programs/word-to-latex/samples/06-Czech%20article%20(UTF8)/camera.xhtml>.

[31] SMIČKA, Radim. *Java Server Pages* [online]. 2005 [cit. 2009-05-11]. Dostupný z WWW: <http://nb.vse.cz/~zelenyj/it380/eseje/xsmir05/jsp.htm>.

[32] STANOI, I., RIEDEWALD, M., AGRAWAL, D., and ABBANDI, A. E. *Discovery of influence sets in frequently updated databases*. In Proceedings of the 27th International Conference on Very Large Data Bases (VLDB 2001), Roma, Italy, September 11-14, 2001, pages 99-108. Morgan Kaufmann.

[33] SUBRAHMANIAN, V. S. *Multimedia database systems*. Morgan Kaufmann. 1998. 442 s. ISBN 1-55860-466-9.

[34] ŠILHAVÁ, J. et al. *Testbench for Evaluation of Image Classifiers.* 2007, no. 9, p. 31-47, ISSN 1811-8992.

[35] ZEZULA, Pavel, et al. *Similarity Search : The Metric Space Approach*. Birkhäuser, 2006. 220 s. ISBN 0387291466.

[36] *PostgreSQL 8.3.7 Documentation: C-Language Functions* [online]. 1996-2009 [cit. 2009-05-11]. Dostupný z WWW: <http://www.postgresql.org/docs/8.3/static/xfunc-c.html>.

[37] *PostgreSQL 8.3.7 Documentation : GIN Indexes* [online]. 1996-2009 [cit. 2009-05-12]. Dostupný z WWW: <http://www.postgresql.org/docs/8.3/static/gin.html>.

[38] Robotics Research Group, University of Oxford. *Affine Covariant Features*. Dostupný z WWW: <http://www.robots.ox.ac.uk/~vgg/research/affine/>.

[39] *TRECVID 2008 topics* [online]. 2008 [cit. 2009-05-12]. Dostupný z WWW: <http://www-nlpir.nist.gov/projects/tv2008/pastdata/topics/topics.2008.xml>.

[40] WIKIPEDIA, Contributors. *Information retrieval* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 5 May 2009 [cit. 2009-05-14]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Information_retrieval&oldid=288143151>.

[41] WIKIPEDIA, Contributors. *Feature detection (computer vision)* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 12 May 2009 [cit. 2009-05-11]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Feature_detection_(computer_vision)&oldid=289484745>.

[42] WIKIPEDIA, Contributors. *GiST* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 20 January 2009 [cit. 2009-05-08]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=GiST&oldid=265237434>.

[43] WIKIPEDIA, Contributors. *Linear discriminant analysis* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 10 May 2009 [cit. 2009-05-11]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Linear_discriminant_analysis&oldid=289052762>.

[44] WIKIPEDIA, Contributors. *Mahalanobis distance* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 1 May 2009 [cit. 2009-04-20]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=Mahalanobis_distance&oldid=287229760>.

[45] WIKIPEDIA, Contributors. *R-tree* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 7 May 2009 [cit. 2009-04-28]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=R-tree&oldid=288550012>.

[46] WIKIPEDIA, Contributors. *B+ tree* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 7 May 2009 [cit. 2009-04-28]. Dostupný z WWW: <http://en.wikipedia.org/w/index.php?title=B%2B_tree&oldid=289827159>.

[47] WIKIPEDIA, Contributors. *Index (databáze)* [online]. Wikipedia, The Free Encyclopedia, 2001-2009 , 28. 04. 2009 [cit. 2009-04-28]. Dostupný z WWW: <http://cs.wikipedia.org/w/index.php?title=Index_(datab%C3%A1ze)&oldid=3899922 >.

[48] RIJSBERGEN, C.J. van. *INFORMATION RETRIEVAL* [online]. 1999 [cit. 2009-01-05]. Dostupný z WWW: <http://www.dcs.gla.ac.uk/~iain/keith/index.htm>.

Seznam příloh

Příloha 1. Přiložené CD

Přiložené CD obsahuje adresáře:

- */aplikace/* - výsledná aplikace implementovaná jako JSP v prostředí NetBeans
- */db/* - schémata tabulek databáze spolu se zálohou dat
- */experimenty/* - kompletní výsledky provedených experimentů
- */pgSiftOrder/* - vzdálenostní funkce implementované v jazyce C
- */text/* - textová zpráva.

Příloha 2. Experimenty

Legenda k tabůlkám

Atributy:

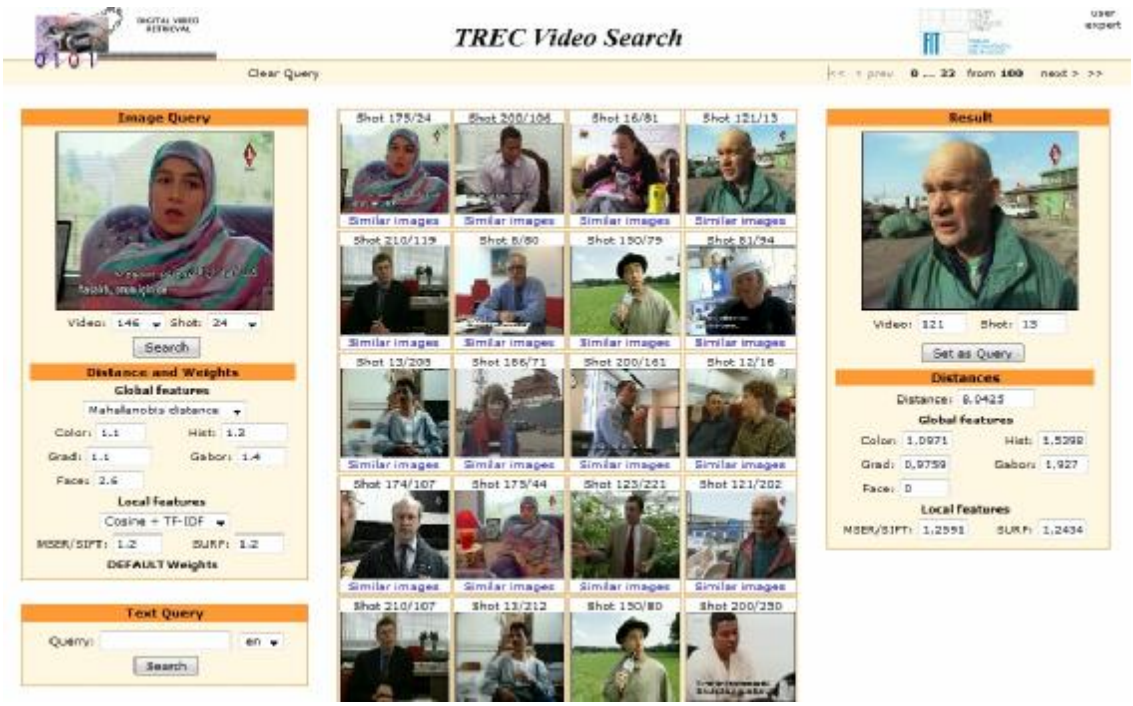
- *col* – barevnost dle specifikace TRECVid
- *hist* – barevnostní histogram
- *grad* – gradienty
- *gabor* – textury
- *face* – obličeje na snímku
- *sift* a *surf*

Vzdálenostní funkce (popsány v sekci 4.2):

- *E* – euklidova
- *CH* – chebyshevova
- *M* – mahalanobisova
- *C* – kosinová
- *B* – binární

Dist64 – vzdálenost 64. snímku od dotazového
Dist100 – vzdálenost 100. snímku od dotazového
Time – doba provádění dotazu v milisekundách
Precision – přesnost
AP – průměrná přesnost úplnosti a přesnosti (stejně jako přesnost popsáno v sekci 2.2)

EXPERIMENT č. 1: Nalezněte snímky s kamerou zaostřující na lidskou tvář.



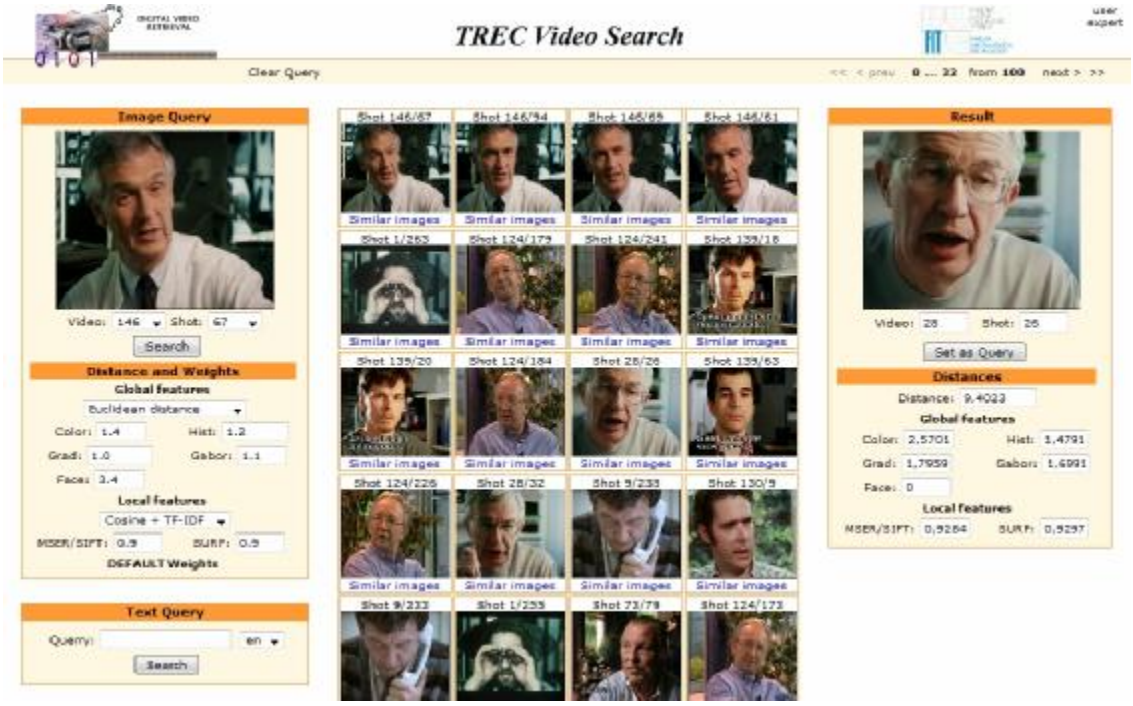
	col E	col CH	col M	hist E	hist CH	hist M	grad E	grad CH	grad M	gabor E	gabor CH	gabor M	face E	face CH	face M	sift C	sift B	surf C	surf B	all
1-4	4	3	4	4	3	3	4	4	3	3	2	3	3	3	3	2	2	3	3	4
5-8	1	2	0	0	0	2	2	2	3	2	3	1	3	3	3	2	1	1	1	4
9-12	2	1	2	2	2	1	1	0	2	2	0	2	4	4	3	2	2	2	0	3
13-16	2	0	1	2	3	2	2	1	2	0	2	3	3	3	2	0	2	0	0	4
17-20	0	2	0	0	2	3	1	0	2	2	0	1	4	4	3	1	0	2	2	4
21-24	2	1	2	1	0	2	2	1	0	2	2	1	3	3	3	2	2	1	0	4
25-28	0	1	0	0	2	1	1	2	1	2	3	3	3	3	4	2	2	1	2	4
29-32	3	1	3	2	1	1	1	3	2	2	2	0	3	3	3	2	2	2	1	3
33-36	1	2	1	1	1	1	2	1	0	3	2	0	2	2	4	0	3	1	0	4
37-40	3	1	1	0	0	1	2	0	0	1	2	2	2	2	4	1	2	2	1	4
41-44	2	0	2	1	0	1	0	0	0	1	2	0	2	2	3	2	0	0	1	4
45-48	0	2	2	1	0	1	0	4	1	3	3	4	3	3	3	0	1	1	1	3
49-52	1	2	2	1	0	1	0	2	1	1	1	3	1	1	3	1	2	1	1	3
53-56	2	0	2	0	0	0	0	2	1	1	2	4	4	4	4	2	3	2	2	2
57-60	2	0	0	2	1	2	1	0	3	3	1	2	4	4	3	1	2	2	1	4
61-64	0	1	1	0	0	2	0	1	2	1	2	1	2	2	4	3	0	2	2	4
All	25	19	23	17	15	24	19	23	23	29	29	30	46	46	52	23	26	23	18	58
Dist64:	116,1	42	65,5	2E+06	8E+05	254	4E+06	1E+06	12,1	39	16	6,6	0	0	0	0,92	289	0,96	291	9,6
Dist100:	119,0	43	67,0	2E+06	8E+05	262,0	4E+06	2E+06	12,5	40,8	17	7,0	0	0	0	0,93	290	0,96	291	
Time (ms)	990	999	1099	1202	1097	2504	941	959	1032	929	947	980	907	909	907	771	770	2899	2622	16927
Precision:	39,1%	29,7%	35,9%	26,6%	23,4%	37,5%	29,7%	35,9%	35,9%	45,3%	45,3%	46,9%	71,9%	71,9%	81,3%	35,9%	40,6%	35,9%	28,1%	90,6%
AP:	21,7%	13,4%	17,8%	13,8%	11,7%	19,1%	18,3%	18,8%	19,4%	23,4%	21,2%	23,5%	55,8%	55,8%	62,3%	14,4%	17,2%	15,5%	10,3%	85,6%

Experiment č. 1

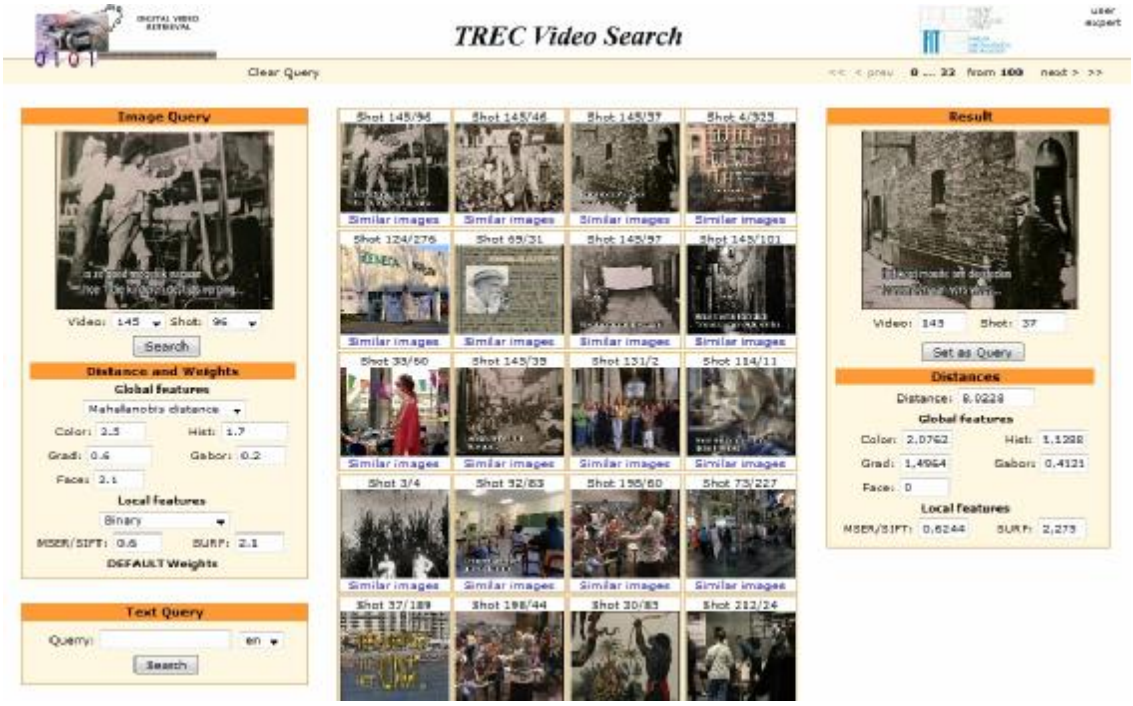
	col E	col CH	col M	hist E	hist CH	hist M	grad E	grad CH	grad M	gabor E	gabor CH	gabor M	face E	face CH	face M	sift C	sift B	surf C	surf B	all
1-4	4	3	4	4	4	4	2	2	2	2	2	3	2	3	3	3	2	3	3	4
5-8	1	1	2	3	2	3	1	2	0	2	3	0	3	3	3	0	0	1	0	3
9-12	2	2	1	2	3	1	1	1	0	0	1	1	3	3	4	0	1	0	1	4
13-16	3	2	1	1	0	0	1	2	1	0	1	0	3	4	4	0	1	1	0	4
17-20	1	1	1	0	3	1	0	1	1	3	1	2	4	4	2	1	0	0	1	3
21-24	1	0	2	0	2	0	1	2	0	2	1	0	2	2	3	0	0	1	1	2
25-28	1	2	3	0	3	2	0	0	1	1	3	0	4	4	4	1	1	1	1	2
29-32	2	0	0	0	1	3	2	1	2	2	0	3	4	4	3	1	0	0	1	3
33-36	0	0	1	2	1	0	1	1	2	0	1	2	4	2	2	0	0	1	0	3
37-40	0	0	1	3	3	0	1	1	2	1	0	0	4	4	3	2	1	0	1	4
41-44	1	1	1	0	2	1	1	1	0	0	2	0	3	1	2	1	1	1	0	4
45-48	1	1	2	2	1	1	2	0	0	1	0	3	2	1	2	1	2	1	0	2
49-52	0	1	0	1	1	0	0	0	1	0	1	0	2	2	3	0	1	0	1	2
53-56	2	0	1	1	3	0	1	0	2	1	0	0	3	3	3	1	1	1	0	3
57-60	1	1	3	0	1	0	0	0	2	0	0	1	4	4	3	1	0	1	0	4
61-64	1	0	1	0	0	0	2	0	1	2	0	0	3	2	3	1	1	1	0	1
All	21	15	24	19	30	16	16	14	17	17	16	15	50	46	47	13	12	13	10	48
Dist64:	132,9	47	74,2	2E+06	9E+05	264,8	4E+06	2E+06	11,2	50,8	20	8,4	0	0	0	0,94	295	0,96	291	8,3
Dist100:	135,8	49	76,2	2E+06	1E+06	271,1	5E+06	2E+06	11,8	52,4	21	8,8	0	0	0	0,95	295	0,96	292	
Time (ms):	979	988	1091	1168	1111	2509	947	960	1028	913	935	982	874	903	882	716	738	1761	1479	17251
Precision:	32,8%	23,4%	37,5%	29,7%	46,9%	25,0%	25,0%	21,9%	26,6%	26,6%	25,0%	23,4%	78,1%	71,9%	73,4%	20,3%	18,8%	20,3%	15,6%	75,0%
AP:	19,0%	11,0%	20,8%	18,6%	30,4%	16,4%	7,5%	9,1%	7,2%	9,9%	11,3%	9,0%	58,4%	56,2%	57,3%	6,6%	4,6%	7,5%	6,3%	63,2%

Experiment č. 2

EXPERIMENT č. 2: Nalezněte snímky muže, mluvícího do kamery v interview točeném uvnitř – nesmí být vidět žádná další osoba.



EXPERIMENT č. 3: Nalezněte snímky jedné nebo více černo-bílých fotografií, vyplňující více než půlku oblasti snímku.



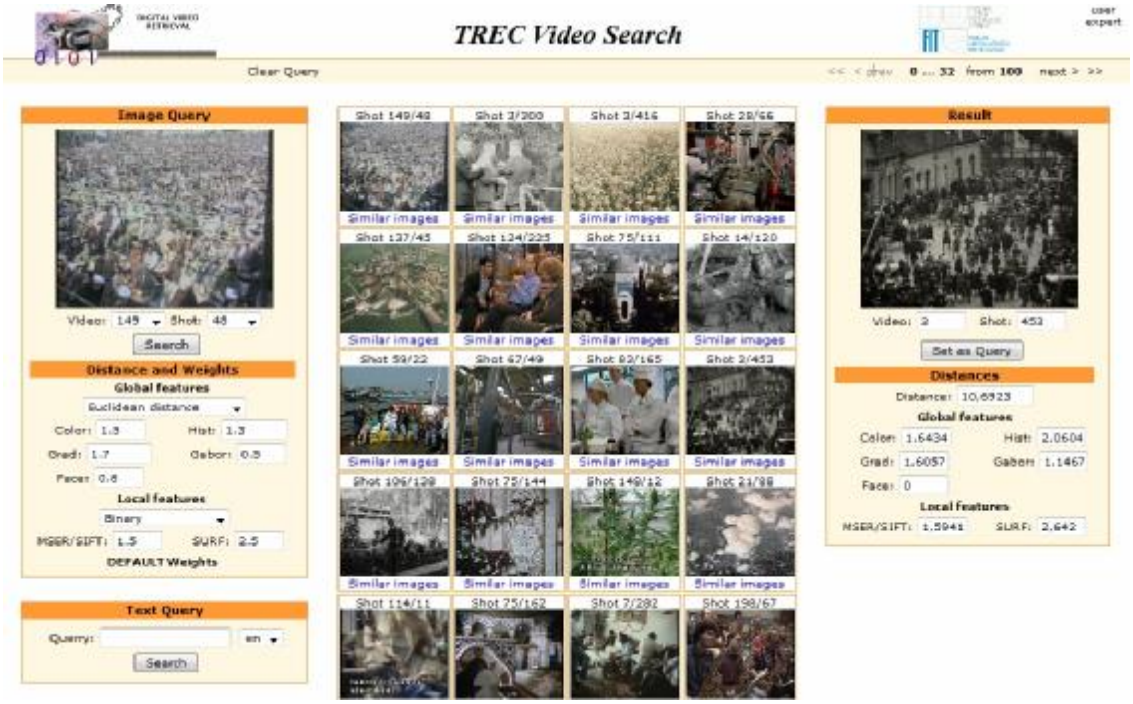
	col E	col CH	col M	hist E	hist CH	hist M	grad E	grad CH	grad M	gabor E	gabor CH	gabor M	face E	face CH	face M	sift C	sift B	surf C	surf B	all
1-4	1	1	1	2	1	1	1	1	1	1	1	1	0	0	0	1	1	2	3	3
5-8	3	1	3	1	1	2	1	1	1	0	0	0	1	1	0	0	0	4	2	1
9-12	3	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	1	1	1
13-16	3	0	4	1	0	2	1	0	0	0	0	0	3	3	3	1	0	0	2	4
17-20	1	0	2	1	1	0	0	0	1	0	0	0	1	2	2	0	0	0	2	0
21-24	1	1	1	0	0	4	0	2	0	0	0	0	1	1	1	0	1	1	1	1
25-28	2	2	1	1	0	1	0	2	0	0	1	0	2	2	2	0	0	0	3	1
29-32	3	1	3	0	0	2	0	0	2	0	0	1	2	2	2	0	1	2	2	1
33-36	2	2	1	1	1	0	0	0	0	1	0	0	2	2	2	0	1	0	3	1
37-40	3	0	3	0	1	1	0	0	0	0	0	0	3	2	2	1	0	1	1	1
41-44	2	1	1	0	1	1	0	0	1	0	0	0	0	0	1	0	0	1	0	1
45-48	2	0	1	3	1	2	0	1	0	0	0	0	2	2	2	0	0	1	1	1
49-52	2	1	2	0	1	1	0	0	1	0	0	0	2	1	1	0	1	0	2	2
53-56	2	0	1	1	0	0	1	0	0	1	0	0	1	1	1	0	1	1	1	2
57-60	3	0	3	0	1	1	0	1	0	0	0	1	1	1	1	1	0	1	1	1
61-64	2	1	2	0	0	2	1	0	0	0	0	0	4	4	4	0	1	0	0	2
All	35	12	30	12	10	21	6	8	7	3	2	3	25	24	24	4	7	15	25	23
Dist64:	82,8	34	49,3	2E+06	7E+05	160,2	6E+06	2E+06	22,0	41,2	18	8,0	0	0	0	0,91	283	0,96	280	9,9
Dist100:	86,0	36	50,4	2E+06	8E+05	166,1	6E+06	2E+06	23,3	46,3	19	8,5	0	0	0	0,92	284	0,96	281	
Time (ms):	992	992	1096	1191	1107	2503	963	985	1051	947	975	997	897	919	920	878	857	4260	2825	9346
Precision:	54,7%	18,8%	46,9%	18,8%	15,6%	32,8%	9,4%	12,5%	10,9%	4,7%	3,1%	4,7%	39,1%	37,5%	37,5%	6,3%	10,9%	23,4%	39,1%	35,9%
AP:	29,4%	4,2%	22,8%	5,8%	3,0%	12,0%	1,8%	2,4%	1,9%	0,6%	0,5%	0,6%	12,6%	12,2%	11,6%	0,8%	1,3%	10,7%	20,6%	16,3%

Experiment č.3

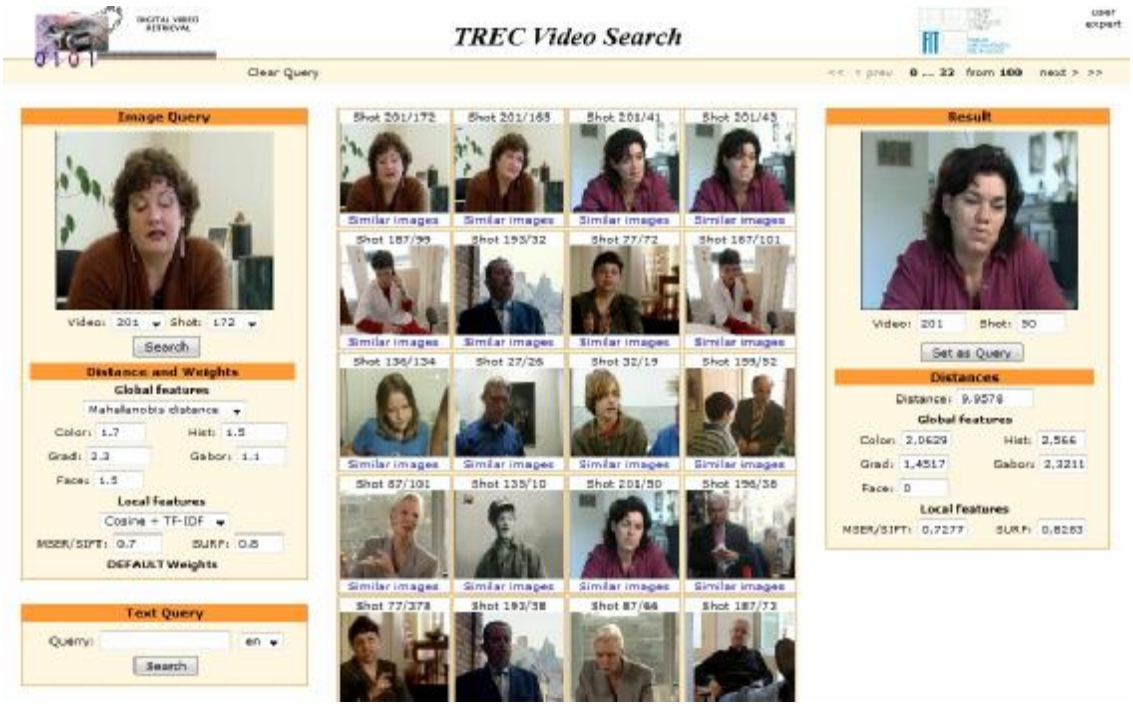
	col E	col CH	col M	hist E	hist CH	hist M	grad E	grad CH	grad M	gabor E	gabor CH	gabor M	face E	face CH	face M	sift C	sift B	surf C	surf B	all
1-4	1	1	1	3	3	3	2	2	1	1	1	1	0	1	0	1	2	3	4	3
5-8	1	1	0	1	0	1	0	1	1	0	0	0	1	0	0	2	1	1	2	1
9-12	1	0	2	0	0	0	1	1	0	0	0	0	0	0	0	0	0	2	4	3
13-16	0	0	0	0	1	0	0	0	3	0	1	0	0	0	0	0	0	1	1	1
17-20	1	0	0	0	0	1	0	1	0	0	0	0	0	1	1	2	0	1	1	2
21-24	1	0	2	0	1	0	0	1	1	0	1	0	0	0	0	1	1	1	2	1
25-28	1	1	1	0	0	1	0	0	1	1	0	0	1	1	1	0	1	1	1	1
29-32	1	1	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1	2	1	1
33-36	1	0	0	2	2	0	1	1	1	0	0	0	0	0	0	0	1	1	0	0
37-40	1	0	1	1	0	0	1	0	1	0	0	0	2	1	1	0	0	1	1	2
41-44	0	1	1	0	1	0	1	0	1	0	0	0	0	0	0	1	0	1	1	1
45-48	0	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	3	1	0	2
49-52	2	1	0	0	1	1	2	1	2	1	0	0	0	0	0	1	1	0	0	2
53-56	0	0	1	2	1	2	1	0	2	0	0	0	0	0	0	0	1	0	1	1
57-60	1	0	0	1	1	1	3	0	1	1	0	0	0	0	0	2	0	2	0	1
61-64	0	0	2	0	1	0	1	0	1	0	0	0	1	0	0	1	0	0	1	1
All	12	7	12	11	13	12	14	10	17	4	3	1	6	5	4	12	12	18	20	23
Dist64:	78,1	35	46,8	2E+06	7E+05	209,2	4E+06	1E+06	13,2	47,1	19	7,6	0	0	0	0,92	268	0,91	260	11,3
Dist100:	80,9	37	48,8	2E+06	7E+05	213,1	4E+06	1E+06	13,8	49,3	20	8,0	0	0	0	0,93	271	0,92	264	
Time (ms):	981	989	1082	1179	1099	2488	936	954	1025	924	935	974	914	891	914	2118	2069	4185	2789	9126
Precision:	18,8%	10,9%	18,8%	17,2%	20,3%	18,8%	21,9%	15,6%	26,6%	6,3%	4,7%	1,6%	9,4%	7,8%	6,3%	18,8%	18,8%	28,1%	31,3%	35,9%
AP:	4,2%	1,7%	4,0%	6,3%	6,8%	6,8%	5,0%	4,7%	6,9%	0,7%	0,8%	0,4%	1,0%	1,1%	0,5%	4,5%	4,9%	12,6%	21,6%	17,2%

Experiment č. 4

EXPERIMENT č. 4: Nalezněte snímky davu lidí pohybujících se venku, zaplňující více než polovinu oblasti snímku.



EXPERIMENT č. 5: Nalezněte snímky ženy, mluvící do kamery v interview točeném uvnitř – nesmí být vidět žádná další osoba.



	col E	col CH	col M	hist E	hist CH	hist M	grad E	grad CH	grad M	gabor E	gabor CH	gabor M	face E	face CH	face M	sift C	sift B	surf C	surf B	all
1-4	3	3	3	3	3	3	4	2	4	3	3	3	1	1	2	2	2	2	2	4
5-8	1	2	0	0	0	0	2	2	3	2	0	1	0	1	0	0	0	1	0	3
9-12	1	0	2	0	1	0	1	0	1	0	2	2	2	1	2	0	0	0	1	2
13-16	0	1	0	0	1	0	0	0	3	2	1	0	1	1	0	0	0	0	0	2
17-20	1	0	0	1	0	0	1	0	1	1	0	0	1	2	1	1	0	0	0	2
21-24	1	1	1	0	1	1	1	0	2	2	1	0	0	0	0	0	0	0	1	3
25-28	3	0	2	1	0	3	1	0	0	0	1	2	0	0	1	1	0	0	0	4
29-32	0	0	2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	3
33-36	2	1	1	1	0	1	0	1	1	0	0	0	1	1	0	0	1	0	0	3
37-40	0	2	0	0	0	0	1	0	0	0	0	0	0	0	2	0	1	2	0	1
41-44	0	0	0	0	3	2	0	1	1	2	1	0	0	0	0	0	0	0	0	2
45-48	1	0	0	3	0	1	0	0	0	0	1	1	1	1	0	0	1	0	0	2
49-52	1	0	1	1	0	0	0	2	0	0	0	0	1	0	0	1	1	0	0	2
53-56	0	1	1	1	0	1	2	2	1	1	0	0	1	1	1	1	0	1	0	1
57-60	1	0	2	0	0	1	0	0	2	0	0	0	1	1	2	0	0	1	0	2
61-64	0	2	0	1	0	0	1	0	1	0	0	1	1	1	2	0	0	0	0	2
All	15	13	15	12	10	13	14	10	20	13	10	10	11	12	13	6	6	7	4	38
Dist64:	121,7	46	68,1	2E+06	8E+05	263,5	4E+06	2E+06	11,3	38,8	17	6,6	0	0	0	0,94	296	0,95	292	11,1
Dist100:	125,1	48	71,7	2E+06	8E+05	271,1	4E+06	2E+06	11,9	40,8	18	7,1	0	0	0	0,95	296	0,96	292	
Time (ms):	992	982	1097	1202	1105	2502	965	1014	1046	943	958	995	896	920	908	682	714	1753	1467	14280
Precision:	23,4%	20,3%	23,4%	18,8%	15,6%	20,3%	21,9%	15,6%	31,3%	20,3%	15,6%	15,6%	17,2%	18,8%	20,3%	9,4%	9,4%	10,9%	6,3%	59,4%
AP:	9,8%	8,6%	9,1%	6,1%	6,3%	7,0%	12,8%	4,6%	20,5%	10,0%	7,0%	7,3%	3,5%	4,3%	5,4%	2,3%	2,2%	2,9%	2,2%	43,0%

Experiment č.5

Příloha 3. Příklad SQL dotazu vyhledávání

```
SELECT kf.dataset AS dataset, kf.video AS video, kf.shot AS shot,
        1.0 * sqrt(distance_square_int4(qr.color, kf.color)) AS color_distance,
        3.0E-5 * sqrt(distance_square_int4(qr.hist, kf.hist)) AS hist_distance,
        3.0E-5 * sqrt(distance_square_int4(qr.grad, kf.grad)) AS grad_distance,
        1.0 * sqrt(distance_square_int4(qr.gabor, kf.gabor)) AS gabor_distance,
        70.0 * sqrt(distance_square_int4(qr.face, kf.face)) AS face_distance,
        1.0 * (1 - rating_cosine_norm(qrl.siftsers, qrl.siftser_weights, qrl.siftser_norm, lf.siftsers,
            lf.siftser_weights, lf.siftser_norm)) AS siftser_rating,
        1.0 * (1 - rating_cosine_norm(qrl.surfs, qrl.surf_weights, qrl.surf_norm, lf.surfs,
            lf.surf_weights, lf.surf_norm)) AS surf_rating,
        ( 1.0 * sqrt(distance_square_int4(qr.color, kf.color)) +
          3.0E-5 * sqrt(distance_square_int4(qr.hist, kf.hist)) +
          3.0E-5 * sqrt(distance_square_int4(qr.grad, kf.grad)) +
          1.0 * sqrt(distance_square_int4(qr.gabor, kf.gabor)) +
         70.0 * sqrt(distance_square_int4(qr.face, kf.face)) +
          1.0 * (1 - rating_cosine_norm(qrl.siftsers, qrl.siftser_weights, qrl.siftser_norm, lf.siftsers,
              lf.siftser_weights, lf.siftser_norm)) +
          1.0 * (1 - rating_cosine_norm(qrl.surfs, qrl.surf_weights, qrl.surf_norm, lf.surfs,
              lf.surf_weights, lf.surf_norm)) + 0 ) AS dist
FROM hlf_search.tv_keyframes AS kf,
        ( SELECT * FROM hlf_search.tv_keyframes_stdev WHERE id=1) AS stdev,
        ( SELECT * FROM hlf_search.tv_keyframes WHERE dataset=821 AND video=161 AND
            shot=27 AND id=1 ) AS qr ,
       hlf_search.tv_localfeatures AS lf,
        ( SELECT * FROM hlf_search.tv_localfeatures WHERE dataset=821 AND video=161
            AND shot=27 AND keyframe=1 ) AS qrl
WHERE kf.dataset=821 AND kf.dataset=lf.dataset AND kf.video=lf.video AND kf.shot=lf.shot
        AND lf.dataset=821 AND qrl.siftsers IS NOT NULL AND qrl.siftser_weights IS NOT
        NULL AND lf.siftsers IS NOT NULL AND lf.siftser_weights IS NOT NULL AND
        qrl.surfs IS NOT NULL AND qrl.surf_weights IS NOT NULL AND lf.surfs IS NOT
        NULL AND lf.surf_weights IS NOT NULL
ORDER BY dist ASC
LIMIT 100
```